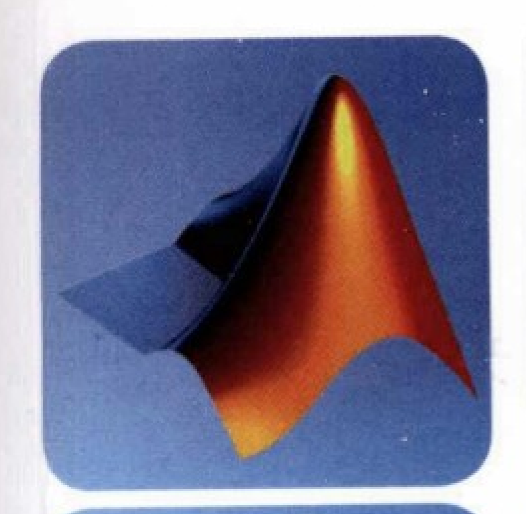


MATLAB 函数速查手册

邓薇 编著









★ **内容全面：**
近500个函数，全面覆盖MATLAB的各类应用

★ **查询方便：**
提供功能索引和字母索引

★ **实例丰富：**
每个函数均配有实例讲解



人民邮电出版社
POSTS & TELECOM PRESS



MATLAB
函数速查手册

本书全面讲解MATLAB各种函数的语法、功能和使用实例，包含以下内容：

MATLAB操作基础
矩阵及其基本运算函数
数值计算函数
符号运算函数
概率统计函数
绘图与图形处理函数
MATLAB程序设计函数
Simulink命令
图形用户界面设计函数
信号处理工具箱函数
符号数学工具箱函数



封面设计：董志桢

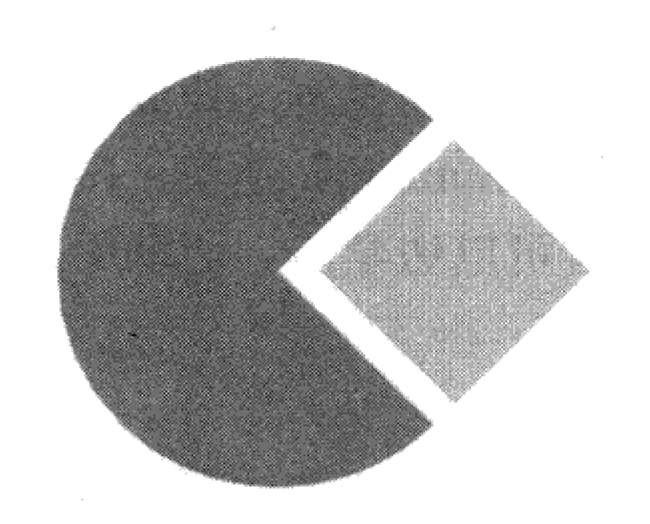
分类建议：计算机/程序设计/MATLAB
人民邮电出版社网址：www.ptpress.com.cn

ISBN 978-7-115-18492-4/TP
定价：49.00 元

内 容 提 要

MATLAB 是目前流行的理论与工程仿真软件之一。该软件自产生以来，就以其独有的特点和明显的优势吸引了各行各业的工作者。本书较全面地介绍了 MATLAB 的函数，主要包括 MATLAB 操作基础、矩阵及其基本运算、与数值计算相关的基本函数、符号运算的函数、概率统计函数、绘图与图形处理函数、MATLAB 程序设计相关函数、Simulink 仿真工具函数、图形用户界面制作函数、信号处理工具箱函数和符号数学工具箱函数等内容。

本书立足 MATLAB 函数基础，并且附带较多的实例讲解，所以既适合初学者，又适合有一定经验的 MATLAB 使用者。本书也可以作为大专院校学生的参考用书。



前　　言

随着科学技术的飞速发展，仿真技术作为一门新兴的学科迅速地登上了历史舞台，无论从实用性的角度还是从缩短开发时间、节约开发资金的角度考虑，任何工程技术方面的研发都离不开仿真，因此，仿真技术越来越受到人们的青睐。不言而喻，MATLAB 这种具有超强功能的仿真软件自然更加受到人们的关注。MathWorks 公司顺应多功能需求的潮流，在其卓越数值计算和图示能力的基础上，又率先在专业水平上开拓了其符号计算、文字处理、可视化建模和实时控制能力，开发了适合多学科要求的新一代科技应用软件 MATLAB。经过多年的国际竞争，MATLAB 已经占据了数值软件市场的主导地位。时至今日，经过不断的完善和改进，MATLAB 7.0 已经得到广泛应用。

本书的特点

1．分类清晰，查询方便

本书按照 MATLAB 函数功能进行分类，读者可以迅速从目录中定位到自己所需要的函数，查看该函数相关内容。本书还将所有函数按首字母排序，方便读者查询。

2．循序渐进，由浅入深

为了方便读者学习，本书首先让读者了解 MATLAB 的产生背

景和功能特点，从读者广为熟悉的数值运算函数出发，逐步深入地介绍符号运算、概率统计等各种函数，配合实例讲解，使读者更容易理解。而后，介绍 MATLAB 程序设计、Simulink 建模、图形用户界面设计以及工具箱函数的使用。读者可以边学习、边动手，更快地掌握 MATLAB 的各种函数。

3．技术全面，内容充实

本书详细讲解了 MATLAB 的各种技术，如基本运算函数、高级应用函数、工具箱等，全书共讲解近 500 个函数，每个函数均用实例加以讲解。

4．实例精讲，深入剖析

根据本人多年的项目经验，MATLAB 仿真工具的应用中，函数的使用尤为重要，任何数值运算、工程上的模型仿真都离不开函数的操作。所以，本书立足于函数，并且配合相关的实例讲解，详细介绍了每个函数的格式、功能以及使用情况。本书选取实用而典型的示例，便于读者参阅和模仿。

本书的内容

本书共 11 章，具体安排如下。

第 1 章：本章将从 MATLAB 是什么，它有哪些特征和优势这些问题入手把读者带到这个仿真软件之中，然后简单介绍了 MATLAB 的运行环境、集成环境以及安装的过程。

第 2 章：本章详细介绍了矩阵运算的基本函数，从矩阵表示、矩阵运算、方程组求解到特征值、线性相关以及稀疏矩阵等，深入讲解了它们的函数格式、使用方法以及应注意的事项。读者要非常熟练地掌握这些函数的使用方法，因为它们是 MATLAB 后续学习的基础。

第 3 章：本章详细介绍了数值运算的函数，从基本数学函数、插值、拟合、数据分析以及数值微积分等几个方面进行了讲解。

第 4 章：本章介绍了符号表达式的一些基本的算术运算，如加、

减、乘、除及合并、分解、展开、化简等，这一系列的操作都有相关的函数以及实例演示，方便读者查阅和练习。

第5章：本章介绍了概率统计相关的函数，包括随机数的产生、随机变量的描述、随机变量的分布函数、数字特征以及参数估计、假设检验等，同时还讲解了图形的绘制，使读者能够对理论知识有直观的理解。

第6章：本章主要讲解MATLAB图形的绘制功能以及图形图像的处理和动画制作。作为一个功能强大的仿真工具软件，MATLAB具有很强的图形处理功能，提供了大量的二维、三维图形函数。由于系统采用面向对象的技术和丰富的矩阵运算，所以在图形处理方面既方便又高效。

第7章：MATLAB不仅是一个功能强大的工具软件，更是一种高效的编程语言。MATLAB软件环境包括了MATLAB语言的编程环境，M文件则是用MATLAB语言编写的程序代码文件。本章从控制流、函数文件、脚本文件以及程序调试等几个方面进行了介绍。

第8章：本章详细介绍了Simulink仿真环境中模型的创建、仿真配置、连续系统建模、离散系统建模及混合系统建模等功能，还讲解了Simulink的仿真原理以及控制系统的设计分析。

第9章：本章主要介绍了图形用户界面设计的基本知识，前一部分主要讲解了图形用户界面设计的基本函数，后一部分以实例的形式介绍了利用控件对界面进行设计的方法。

第10章：本章介绍了工程中较为常用的信号处理工具箱，以实例的形式讲解了信号产生、时频分析以及滤波器的设计函数。

第11章：本章介绍了符号数学工具箱的使用，MATLAB符号运算是通过集成在MATLAB中的符号数学工具箱来实现的。和其余的工具箱有所不同，该工具箱不是基于矩阵的数值分析，而是使用字符串来进行符号分析与运算。MATLAB的符号数学工具箱可以完成几乎所有符号运算功能。其中有符号表达式运算，符号表达式的复合、化简，符号矩阵的运算，符号微积分，符号函数画图，符

号微分方程求解等。

本书在所有的章节中都安排了实例，这些实例具有很强的针对性和代表性，帮助读者能够轻松地掌握函数。

适合的读者

- 工程仿真设计人员
- 数学理论计算人员
- 工程研发人员
- 大、中专院校的学生
- 社会培训学生

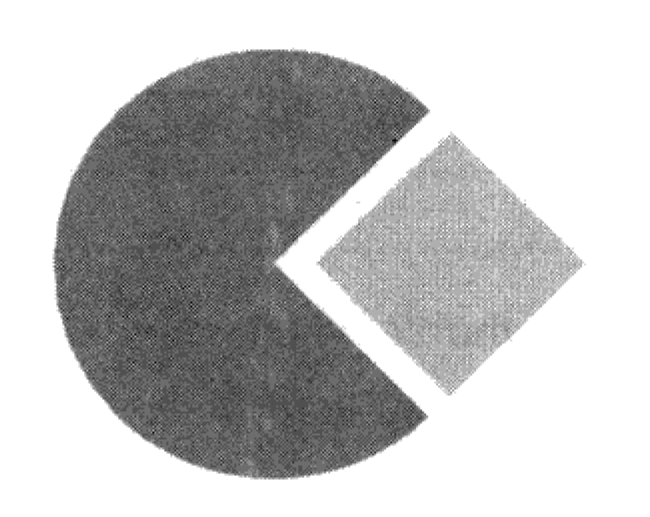
编者与致谢

本书由邓微主持编写，其他参与编写和资料整理的人员有曹秩倩、陈轮、董世星、陈能技、陈向辉、宫垂刚、陈鑫玮、陈垚光、程高伟、戴敏梅、邓尉、董国栋、董霖、段毅、方擎、高德波、龚小鹏、陈衍卿、韩雷、郝红旗、何俊斌、陈东、贺文婧、侯利军、胡诗群、胡添、扈新波、常建功、华剑锋、黄洁娴、博奎、孙健等。在此对大家的辛勤工作表示感谢！

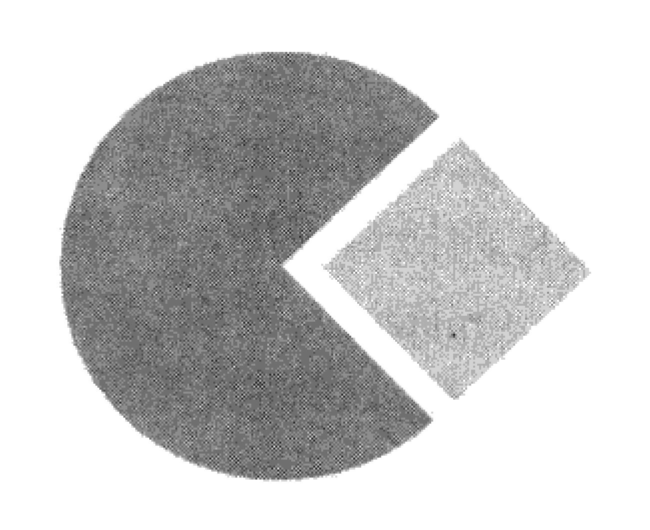
本书责任编辑的联系方式为huangyan@ptpress.com.cn，欢迎来信交流。

编者

2008.8



目　录



第1章　MATLAB 操作基础

MATLAB 作为一种强大的数值计算仿真工具，目前应用得越来越广泛，无论是从理论的角度还是从工程应用的层次，MATLAB 都起到了极为关键的作用。本章为这本书的第 1 章，主要讲述 MATLAB 的一些基础性知识，包括发展背景、主要功能、语言特点、运行环境、安装过程和集成环境等几个方面，最后给出几个简单的例子让读者对 MATLAB 有一个感性的认识。

1.1　MATLAB 概述

经过近 20 年的实践，人们已经意识到：MATLAB 作为计算工具和科技资源，可以扩大科学研究的范围、提高工程生产的效率、缩短开发周期、加快探索步伐、激发创造活力。那么 MATLAB 发展到今天经历了怎样的历程？作为当前最新版本的 MATLAB 7.0 包括哪些内容以及具有哪些功能呢？下面逐一介绍。

1.1.1　MATLAB 产生的历史背景

在 20 世纪 70 年代中期，Cleve Moler 博士和其同事开发了调用

EISPACK 和 LINPACK 的 FORTRAN 子程序库。EISPACK 是特征值求解的 FOETRAN 程序库，LINPACK 是解线性方程的程序库。当时，这两个程序库代表矩阵运算的最高水平。

20 世纪 70 年代后期，身为美国 New Mexico 大学计算机系系主任的 Cleve Moler 编写了 EISPACK 和 LINPACK 的接口程序，并给这个接口程序取名为 MATLAB。该名为矩阵（matrix）和实验室（labotatory）两个英文单词的前 3 个字母的组合。在以后的数年里，MATLAB 在多所大学里作为教学辅助软件使用，并作为面向大众的免费软件广为流传。

1983 年，MATLAB 深深地吸引了工程师 John Little。John Little 敏锐地觉察到 MATLAB 在工程领域的广阔前景。同年，他和 Cleve Moler、Steve Bangert 一起，用 C 语言开发了第二代专业版。这一代的 MATLAB 语言同时具备了数值计算和数据图示化的功能。1984 年，正式把 MATLAB 推向市场，并继续进行 MATLAB 的研究和开发。时至今日，经过 MathWorks 公司的不断完善，MATLAB 7.0 目前已经得到广泛的应用。MATLAB 已经发展成为适合多学科、多种工作平台的功能强大的大型软件。

1.1.2 MATLAB 的主要功能

MATLAB 包括命令控制、可编程，有上百个预先定义好的命令和函数。这些函数能通过用户自定义函数进一步扩展。MATLAB 有许多强有力的命令。例如，MATLAB 能够用一个单一的命令求解线性系统，能完成大量的高级矩阵处理；MATLAB 有强有力的二维、三维图形工具；MATLAB 能与其他程序一起使用。根据 MATLAB 可以实现的任务性质，将其强大的功能划分为如下几个方面。

- 数值计算和符号计算功能：MATLAB 以矩阵作为数据操作的基本单位，还提供了十分丰富的数值计算函数。MATLAB 和著名的符号计算语言 Maple 相结合，使得 MATLAB 具有符号计算功能。

- 绘图功能：MATLAB 提供了两个层次的绘图操作，一种是对图形句柄进行的低层绘图操作，另一种是建立在低层绘图操作之上的高层绘图操作。
- 编程语言：MATLAB 具有程序结构控制、函数调用、数据结构、输入输出、面向对象等程序语言特征，而且简单易学、编程效率高。
- MATLAB 工具箱：MATLAB 包含两部分内容，基本部分和各种可选的工具箱。MATLAB 工具箱分为两大类：功能性工具箱和学科性工具箱。

25 个不同的 MATLAB 工具箱可应用于特殊的应用领域。MATLAB 是一个十分有效的工具，在以下的领域里可解决各种问题。

- 工业研究与开发。
- 数学教学，特别是线性代数。线性代数中几乎所有基本概念都能涉及并能够得到很好的解答。
- 在数值分析和科学计算方面的教学与研究。能够详细地研究和比较各种算法。
- 在诸如电子学、控制理论和物理学等工程和科学学科方面的教学与研究。
- 在诸如经济学、化学和生物学等有计算问题的所有其他领域中的教学与研究。
- 在 MATLAB 中创建的组是矩阵。

1.1.3 MATLAB 的语言特点

被称作第四代计算机语言的 MATLAB，利用其丰富的函数资源，使编程人员从繁琐的程序代码中解放出来。MATLAB 最突出的特点就是简洁，它给用户带来的是最直观、最简洁的程序开发环境。以下简单介绍一下 MATLAB 的主要特点。

- 语言简洁紧凑，使用方便灵活，库函数极其丰富。MATLAB 程序书写形式自由，库函数都由本领域的专家编写，用户不必担心

函数的可靠性。

- 运算符丰富。MATLAB 是用 C 语言编写的，MATLAB 提供了和 C 语言几乎一样多的运算符，灵活使用 MATLAB 的运算符将使程序变得极为简短。
- MATLAB 既具有结构化的控制语句（如 for 循环、while 循环、break 语句和 if 语句），又有面向对象编程的特性。
- 程序的可移植性很好，基本上不做修改就可以在各种类型的计算机和操作系统上运行。
- MATLAB 的图形功能强大。在 FORTRAN 和 C 语言里，绘图都很不容易，但在MATLAB里，数据的可视化非常简单。MATLAB还具有较强的编辑图形界面的能力。
- MATLAB 的缺点是它和其他高级程序相比，程序的执行速度较慢。这是因为 MATLAB 的程序不用编译等预处理，也不生成可执行文件，程序为解释执行。
- 功能强大的工具箱是 MATLAB 的另一特色。MATLAB 包含两个部分：核心部分和各种可选的工具箱。核心部分中有数百个核心内部函数。其工具箱又分为两类：功能性工具箱和学科性工具箱。功能性工具箱主要用来扩充其符号计算功能、图示建模仿真功能、文字处理功能以及与硬件实时交互功能。功能性工具箱用于多种学科。
- 源程序的开放性。除内部函数以外，所有 MATLAB 的核心文件和工具箱文件都是可读可改的源文件，用户可通过对源文件的修改以及加入自己的文件构成新的工具箱。

1.2 MATLAB 的运行环境及安装

1.2.1 MATLAB 的运行环境

MATLAB 可以安装到下列各种类型的计算机上：PC 及兼容机、

Macintosh 机、Sun 工作站、VAX 机、HP 工作站、Apollo 工作站等。而且无论是单机还是网络环境都可发挥其卓越的性能。如果单纯地使用 MATLAB 语言进行编程，不连接其他外部语言，则用 MATLAB 语言编写出来的程序可以不做任何修改地移植到其他机型上使用，所以 MATLAB 和其他语言不同，它是和计算机类型无关的。本书仅介绍在 PC 机环境下 MATLAB 的应用。MATLAB 7.0 对 PC 机的系统要求如下：

- 操作系统为 Microsoft Windows 98/ NT/2000 或 Windows XP；
- Intel 480 以上 CPU，建议使用奔腾处理器；
- 光驱，用来安装 MATLAB；
- 最小 16MB 内存；
- 8 位以上显卡；
- 建议安装 Internet Explorer 4.0 浏览器，方便使用 MATLAB 帮助文件；
- 建议安装 Adobe Acrobat Reader，用来阅读和打印 MATLAB 中 PDF 格式的在线帮助文件；
- 建议安装 Microsoft Word，用来运行 MATLAB Netebook。

1.2.2 MATLAB 7.0 的安装

合理正确地安装是一个程序运行和工作的基础，在这个基础上才能讨论其他有关的操作。在这个小节中将介绍与 MATLAB 7.0 安装相关的内容，这个详细的过程将会指导初学者顺利完成安装过程。

（1）把 MATLAB 7.0 的安装盘放入计算机，或者将安装程序复制到要安装的计算机上。找到安装程序（通常命名为“setup.exe”），运行该程序，会出现如图 1.1 所示的界面。选择“Install”单选项，单击“Next”按钮，出现如图 1.2 所示界面。

（2）设置用户相关信息。单击“Next”按钮，得到如图 1.3 所示的图形界面。

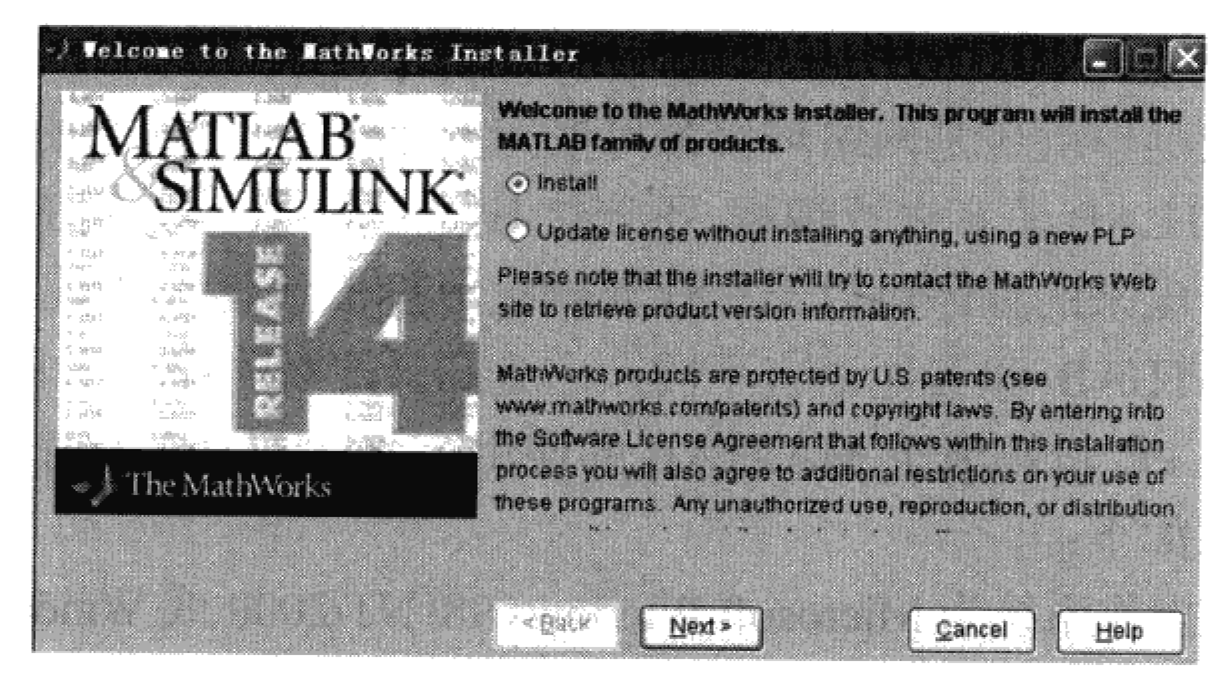


图 1.1 MATLAB 7.0 安装界面

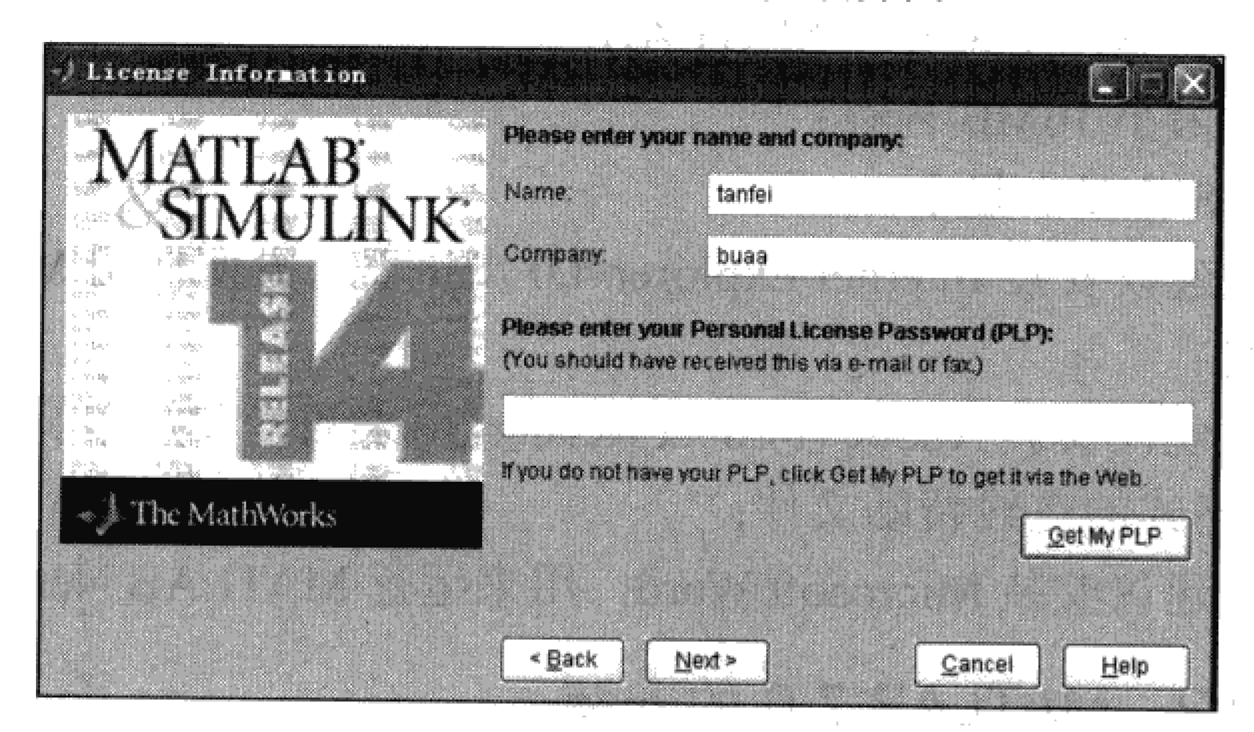


图 1.2 MATLAB 7.0 的设置

License Agreement

The MathWorks, Inc.
Software License

IMPORTANT NOTICE

THE LICENSE AGREEMENT TOGETHER WITH ANY APPLICABLE ADDENDUM REPRESENTS THE ENTIRE AGREEMENT BETWEEN YOU (THE "LICENSEE") AND THE MATHWORKS, INC. ("MATHWORKS") CONCERNING THE PROGRAM(S) AND DOCUMENTATION DELIVERED HEREUNDER.

Do you accept the terms of the license agreement? Yes No

< Back Next > Cancel Help

图 1.3 MATLAB 7.0 安装条款

（3）在图1.3中，显示了MATLAB 7.0相关条款，只有接受了这个条款的相关约定，才能进行后面的安装工作。选择“Yes”单选项，“Next”按钮被激活，单击此按钮，进入如图1.4所示界面。

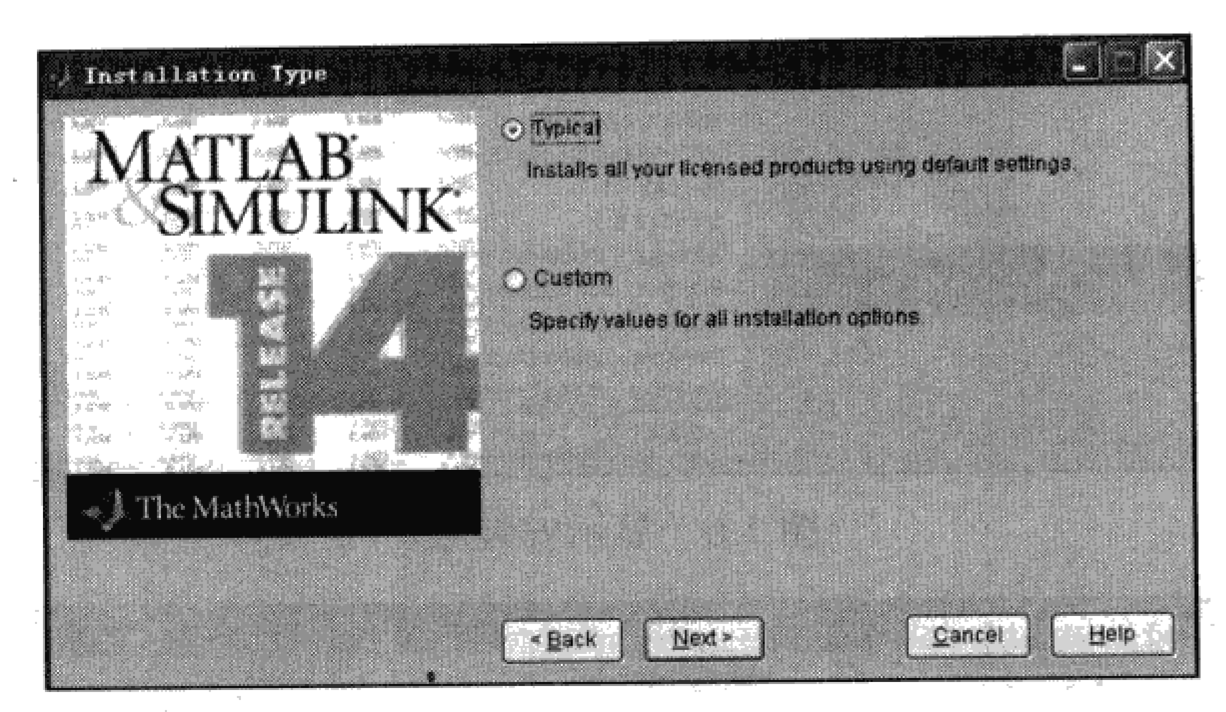


图1.4 选择安装类型

（4）在该界面中选择安装的类型，本例中选择为典型安装（Typical），单击“Next”按钮，打开如图1.5所示界面。

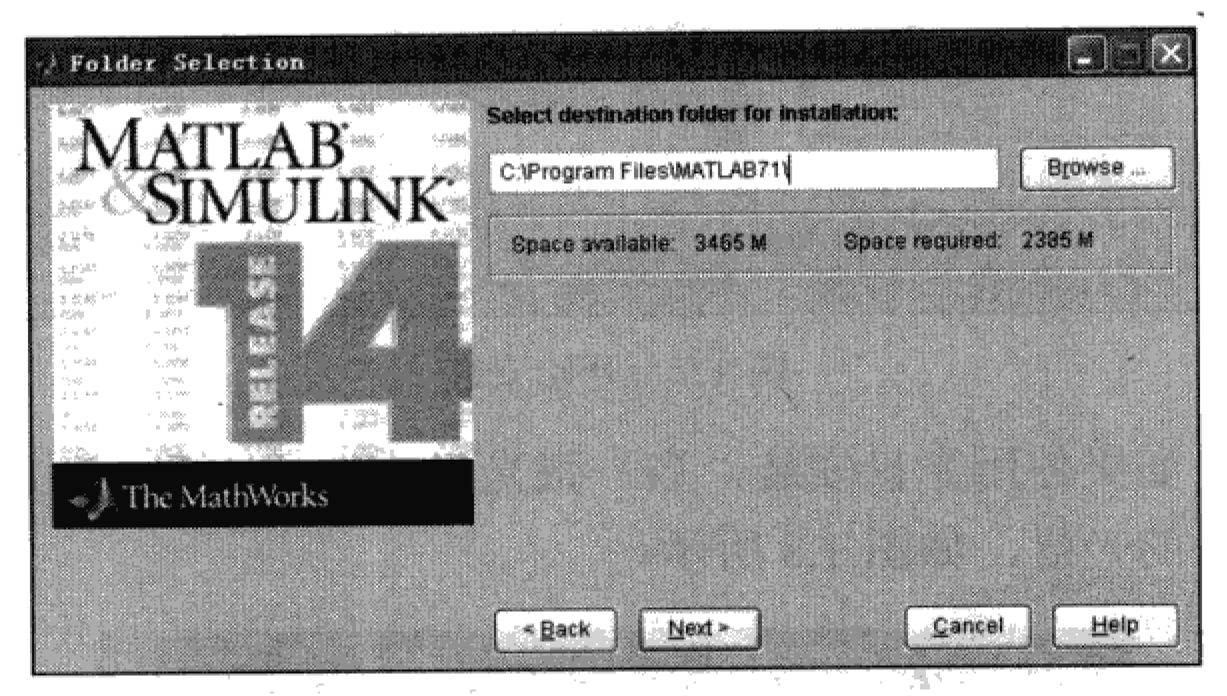


图1.5 设置安装路径

（5）在该界面中设置安装的路径，设置完后单击“Next”按钮，打开如图1.6所示界面。

（6）在该界面中显示所选择的安装选项，单击“Install”按钮弹出如图1.7所示界面开始安装。

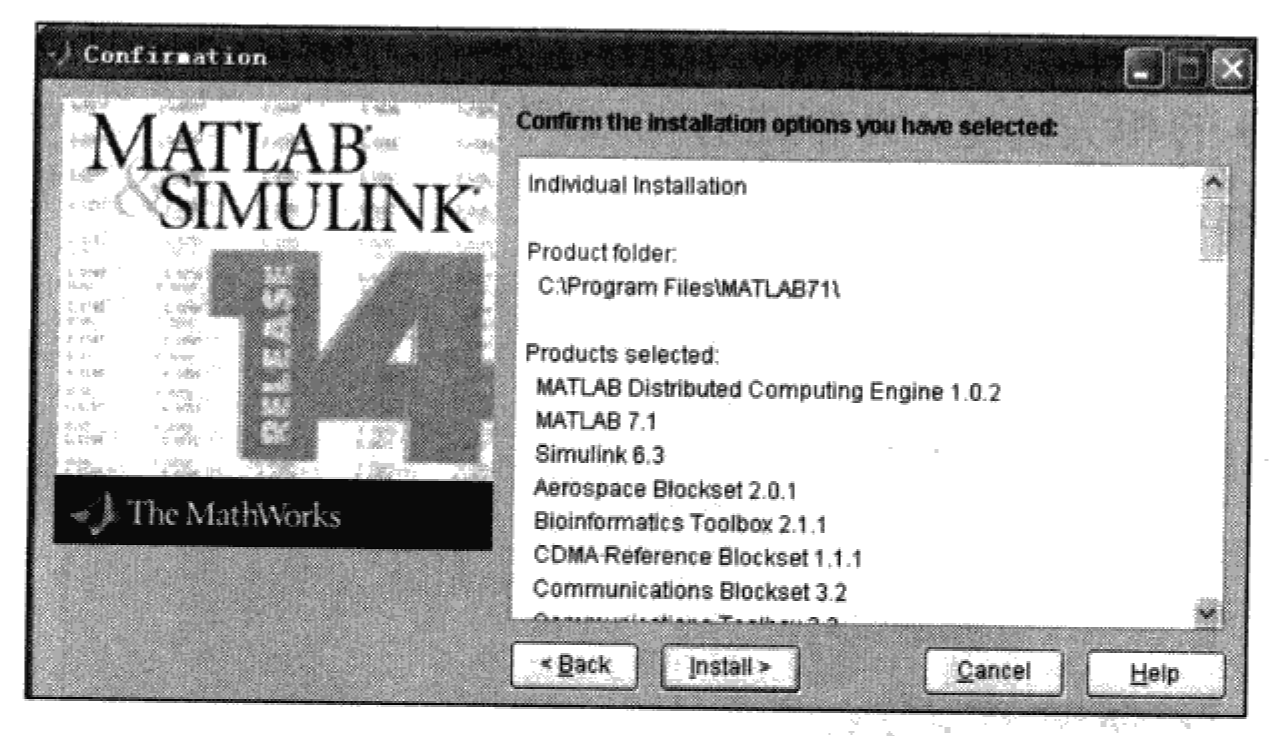


图 1.6　显示安装项

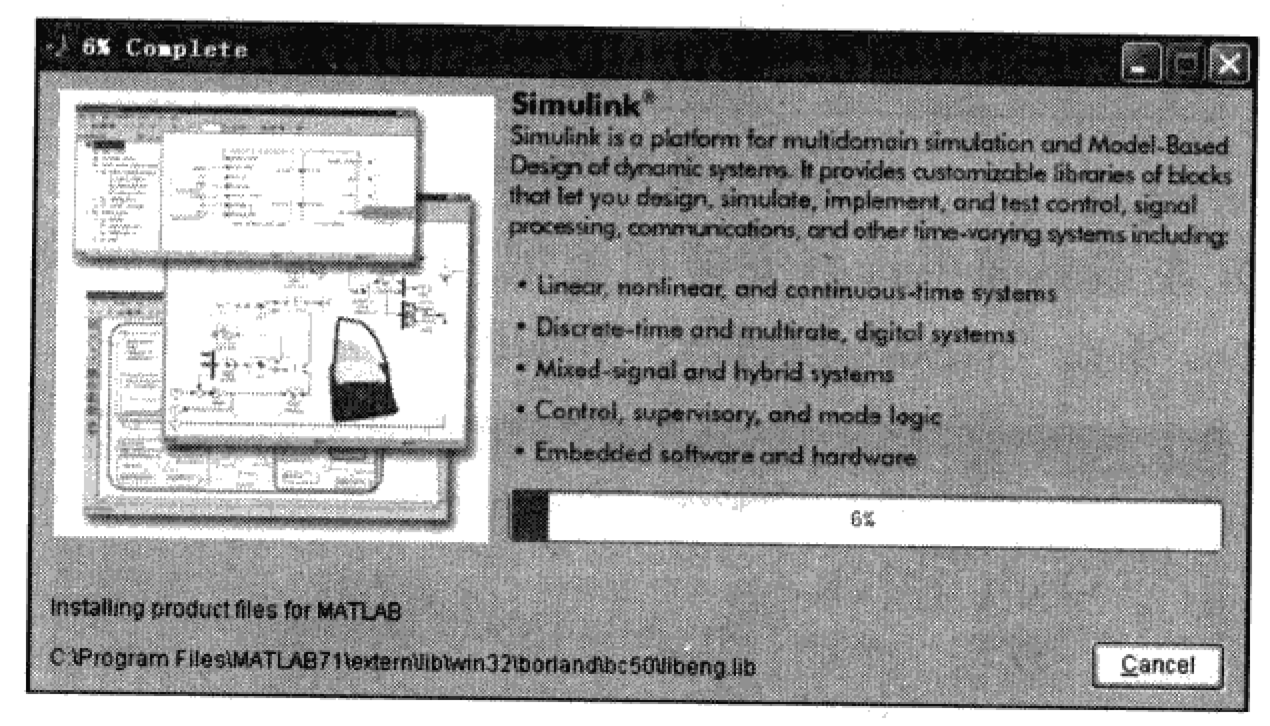


图 1.7　开始安装

（7）如果使用光盘安装，当一张光盘安装完成后，中间会提醒插入另外的光盘，如图 1.8 所示。

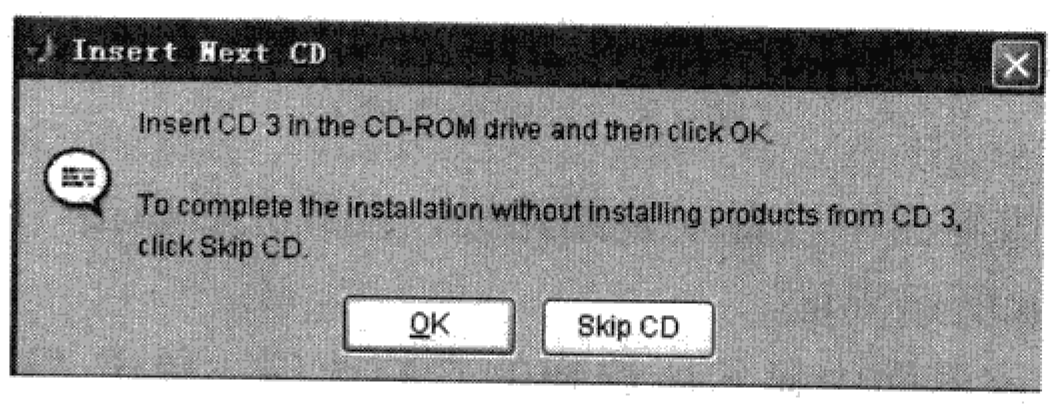


图 1.8　插入另外一张光盘

（8）单击“OK”按钮继续安装，安装完成后界面如图1.9所示。单击“Finish”按钮完成MATLAB的安装。

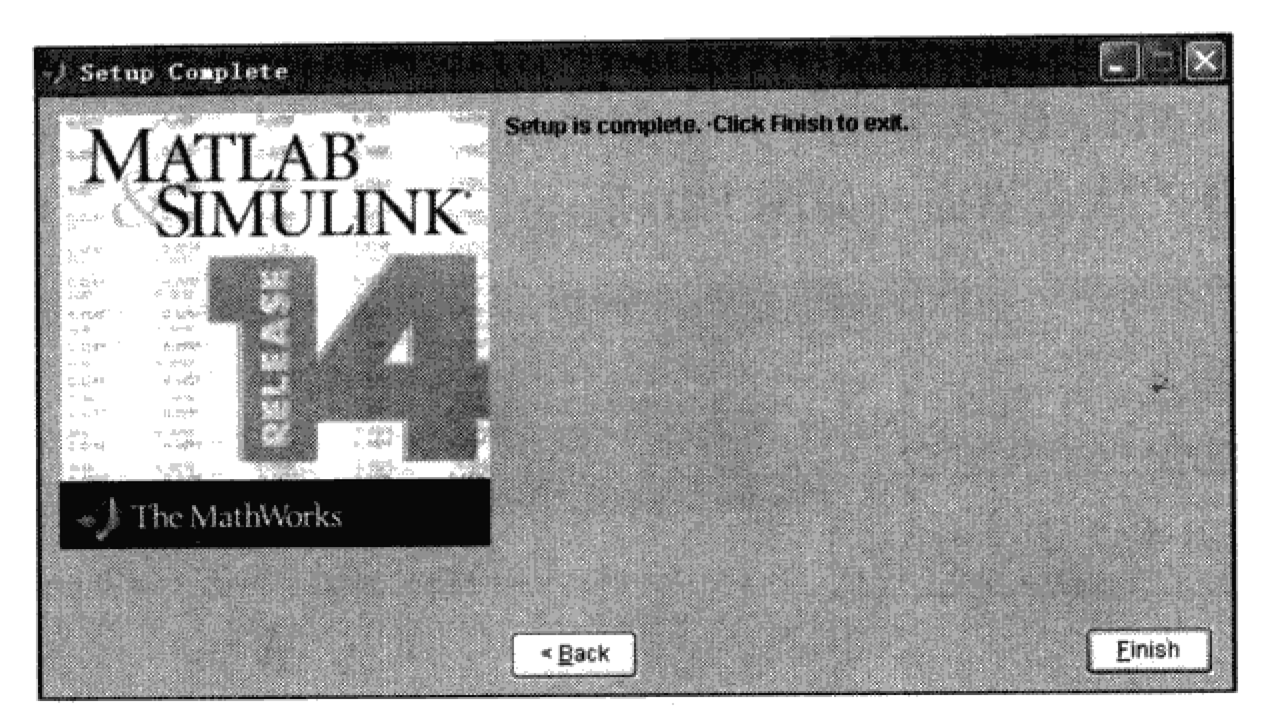


图1.9　安装完成

1.3　MATLAB集成环境

MATLAB 7.0是一个高度集成的语言环境，在该环境下既可以进行交互式的操作，又可以编写程序、运行程序并跟踪调试程序。

1.3.1　启动与退出MATLAB集成环境

MATLAB系统的启动与一般的Windows程序一样，启动MATLAB系统有3种常用方法。

- 选择“开始”|“程序”|“MATLAB”|“MATLAB”命令启动MATLAB。
- 利用Windows建立快捷方式的功能，将MATLAB程序的快捷方式放在桌面上。只要在桌面上双击该图标即可启动MATLAB。
- 运行matlab.exe文件启动MATLAB。

启动MATLAB后，将进入MATLAB 7.0集成环境。MATLAB 7.0

集成环境包括 MATLAB 主窗口、命令窗口（Command Window）、工作空间窗口（Workspace）、命令历史窗口（Command History）、当前目录窗口（Current Directory）和启动平台窗口（Launch Pad）。

当 MATLAB 安装完毕并首次启动时，展现在屏幕上的界面为 MATLAB 的默认界面，如图 1.10 所示。

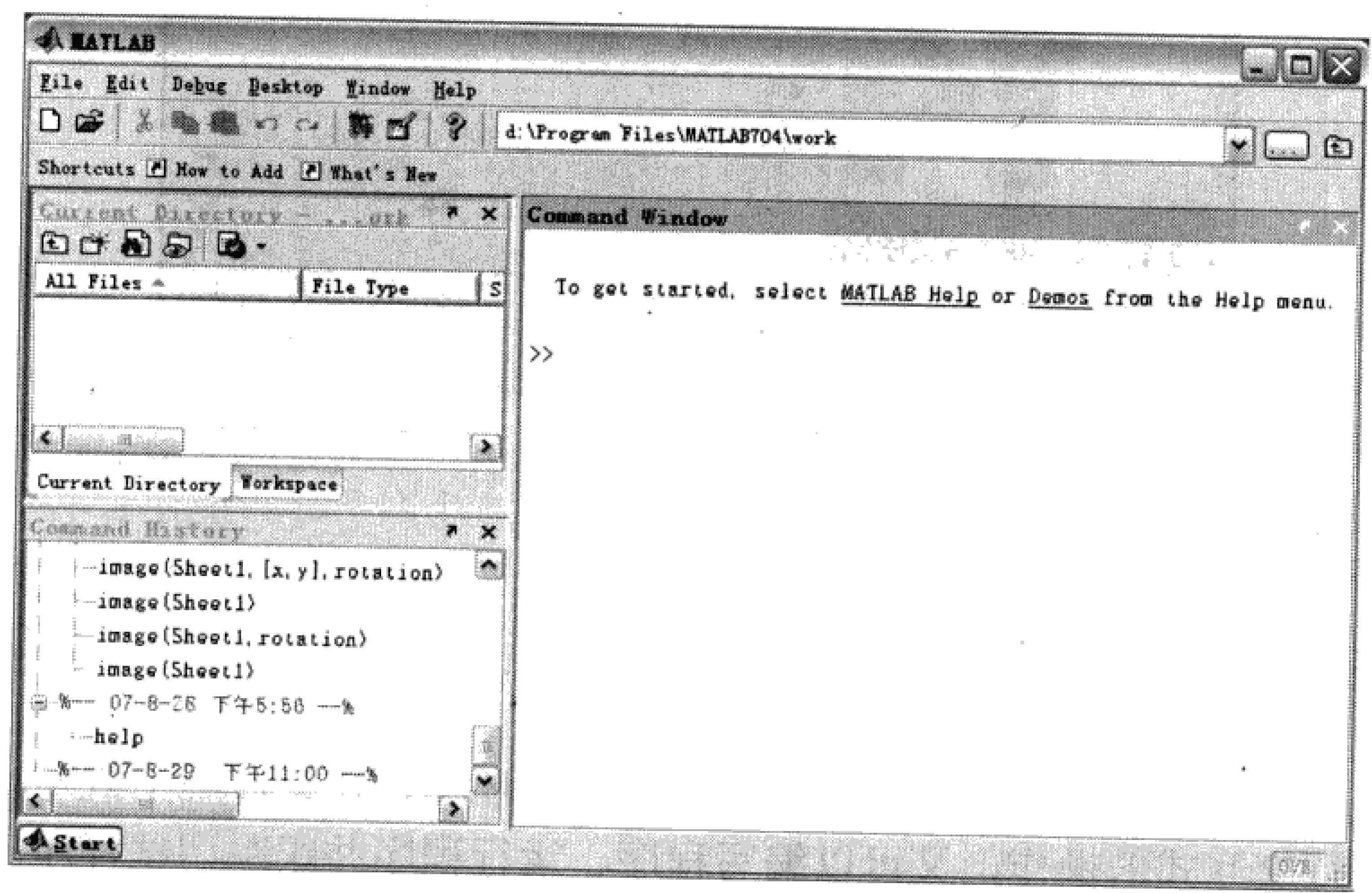


图 1.10　MATLAB 7.0 默认初始界面

要退出 MATLAB 系统，也有 3 种方法：

- 单击 MATLAB 命令窗口中的“关闭”按钮；
- 在 File 菜单中选择“Exit”命令；
- 在 MATLAB 命令窗口中输入“Exit”和“Quit”命令。

在 MATLAB 中，变量名是以字母开头，后接字母、数字或下划线的字符序列，最多 19 个字符，且区分字母的大小写。MATLAB 提供的标准函数名必须用小写字母。

1.3.2　MATLAB 的命令窗口

命令窗口是 MATLAB 的主要交互窗口，用于输入命令并显示

除图形以外的所有执行结果，其界面如图 1.11 所示。

```
Command Window

  To get started, select MATLAB Help or Demos from the Help menu.

>> A = 100

A =

   100

>> b = sin(3.245)

b =

   -0.1032

>> C = [1,2,3;4,5,6;7,8,9;3,7,9]

C =

     1     2     3
     4     5     6
     7     8     9
     3     7     9
```

图 1.11 MATLAB 命令窗口

图 1.11 中光标所在的空白区域为命令编辑区，命令编辑区用于输入命令和显示计算结果。键入命令并按下回车键后，MATLAB 就会解释执行所输入的命令，并在命令后面给出计算结果。

MATLAB 命令窗口中的“>>”为命令提示符，表示 MATLAB 正处在准备状态。在命令提示符后键入命令并按下回车键后，MATLAB 就会解释执行所输入的命令，并在命令后面给出计算结果。一般来说，一个命令行输入一条命令，以回车结束。但一个命令行也可以输入若干条命令，各命令之间以逗号分隔，若前一命令后带有分号，则逗号可以省略。例如：

```
a=10,b=20
a=15;b=30
```

如果一个命令行很长，一个物理行之内写不下，可以在第一个物理行之后加上 3 个小黑点并按下回车键，然后在下一个物理行中继续写命令的其他部分。3 个小黑点称为续行符，即把下面的物理

行看作该行的逻辑继续。

1.3.3 工作空间窗口

工作空间是 MATLAB 用于存储各种变量和结果的空间。在该窗口中显示工作空间中所有变量的名称、字节数和变量类型，可对变量进行观察、编辑、保存和删除等操作。其界面如图 1.12 所示。

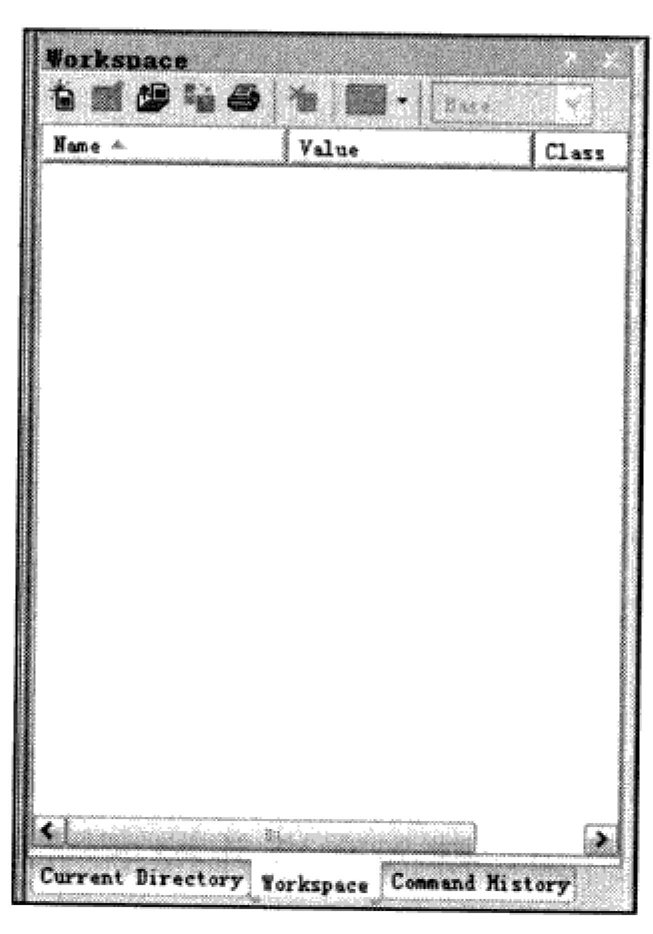


图 1.12　工作空间窗口

1.3.4 当前目录窗口

当前目录是指 MATLAB 运行文件时的工作目录，只有在当前目录或搜索路径下的文件、函数可以被运行或调用。在当前目录窗口中可以显示或改变当前目录，还可以显示当前目录下的文件并提供搜索功能。将用户目录设置成当前目录也可使用 cd 命令。例如，将用户目录 c:\mydir 设置为当前目录，可在命令窗口中输入如下命令：

```
cd c:\mydir
```

MATLAB 的当前目录窗口如图 1.13 所示。

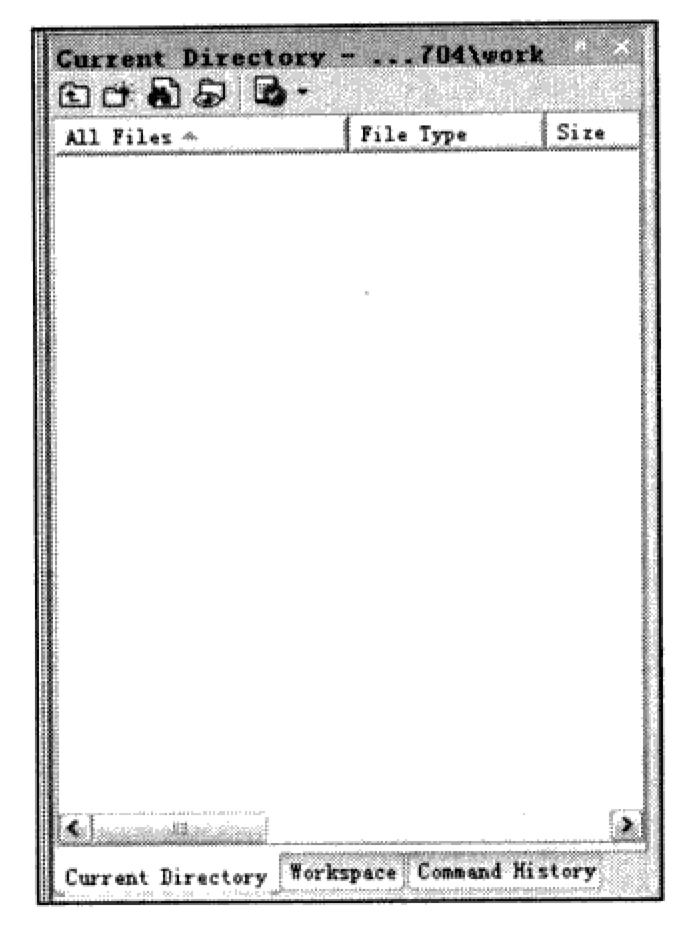


图 1.13 当前目录窗口

1.3.5 MATLAB 的搜索路径

当用户在 MATLAB 命令窗口中输入一条命令后，MATLAB 按照一定次序寻找相关的文件。基本的搜索过程为：

（1）检查该命令是不是一个变量；

（2）检查该命令是不是一个内部函数；

（3）检查该命令是否是当前目录下的 M 文件；

（4）检查该命令是否是 MATLAB 搜索路径中其他目录下的 M 文件。

用户可以将自己的工作目录列入 MATLAB 搜索路径，从而将用户目录纳入 MATLAB 系统统一管理。设置搜索路径的方法如下。

- 用 path 命令设置搜索路径。例如，将用户目录 c:\mydir 加到搜索路径下，可在命令窗口中输入命令：path（path,'c:\mydir'）。
- 用对话框设置搜索路径。在 MATLAB 的 File 菜单中选择“Set Path”命令或在命令窗口中执行“pathtool”命令，将出现“搜索路径设置”对话框。通过“Add Folder”或“Add with Subfolder”

命令按钮将指定路径添加到搜索路径列表中。在修改完搜索路径后，则需要保存搜索路径。

1.3.6 命令历史记录窗口

在默认设置下，历史记录窗口中会自动保留自安装起所有用过的命令的历史记录，并且还标明了使用时间，从而方便用户查询。而且，通过双击命令可进行历史命令的再运行。如果要清除这些历史记录，可以选择 Edit 菜单中的“Clear Command History”命令。MATLAB 的命令历史记录窗口如图 1.14 所示。

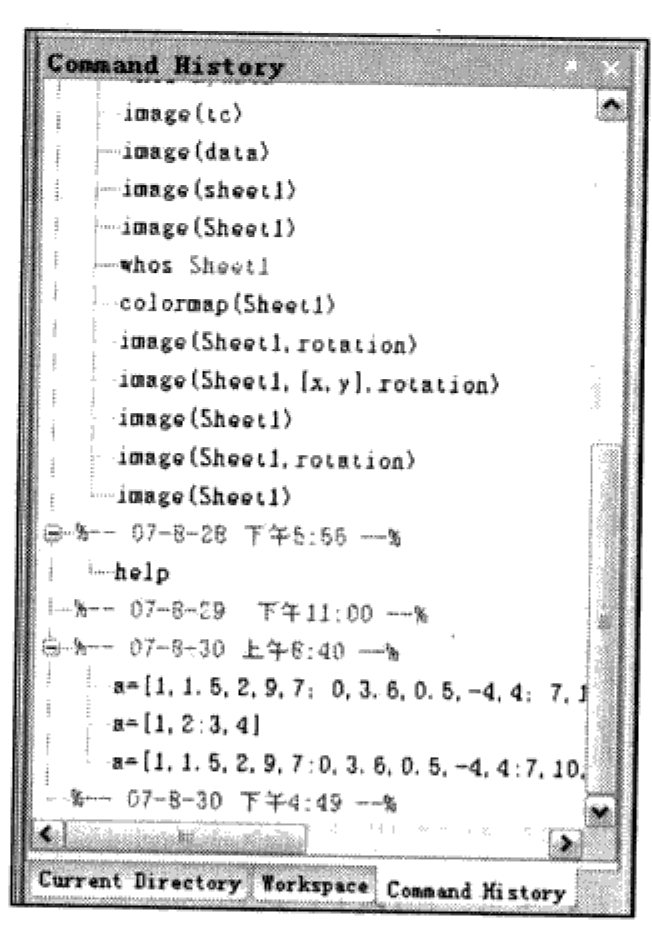


图 1.14 命令历史记录窗口

1.3.7 启动平台窗口和 Start 按钮

在 MATLAB 7.0 的启动平台窗口中用户可以方便地打开和调用 MATLAB 的各种程序、函数和帮助文件。

MATLAB 7.0 主窗口左下角还有一个 Start 按钮，单击该按钮会弹出一个菜单，选择其中的命令可以执行 MATLAB 产品的各种工具，并且可以查阅 MATLAB 包含的各种资源。如图 1.15 所示为

MATLAB 中与 Simulink 相关的工具命令。MATLAB 中工具箱的主要命令和资源信息如图 1.16 所示。

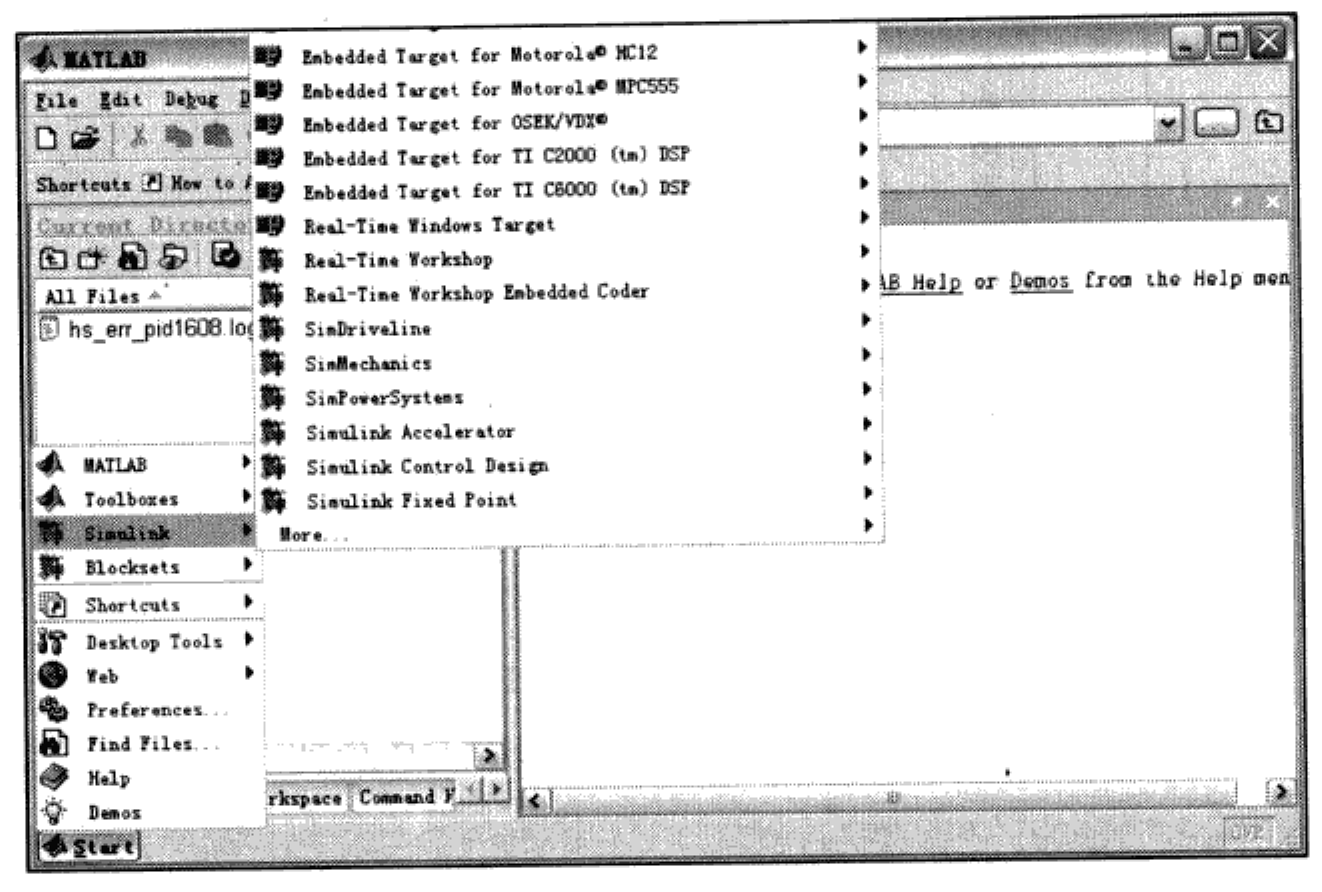


图 1.15 Start 按钮之 Simulink

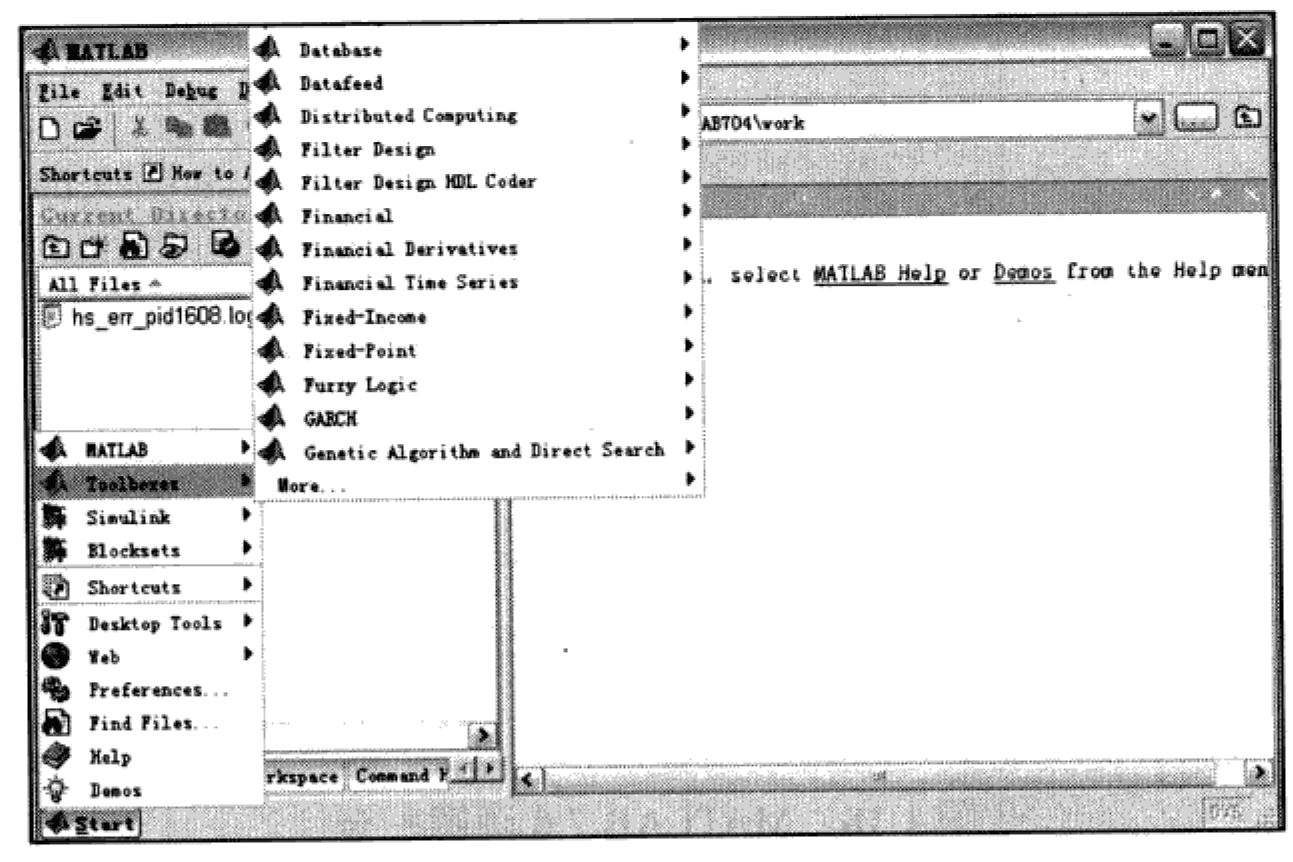


图 1.16 Start 按钮之工具箱

其他的命令操作方式与上面介绍的两种相同，需要使用 MATLAB 中的哪些工具或函数就按照上面的过程逐次选取执行。如图 1.17 所

示为使用画图平台的过程，按照图中变灰的顺序依次选择，最后单击 “Plot Tools” 命令，即可打开如图 1.18 所示的画图平台。

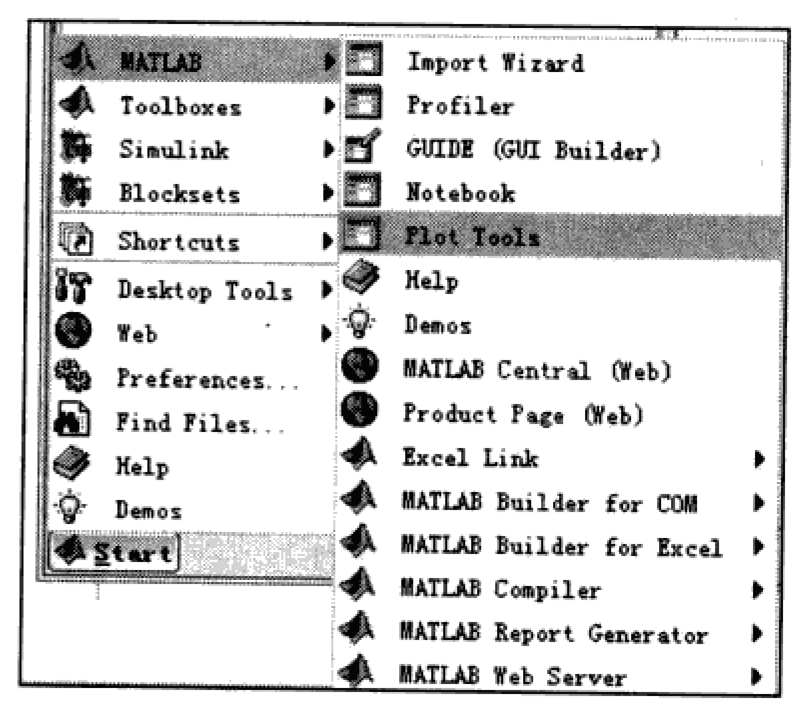


图 1.17　使用 MATLAB 7.0 Start 按钮

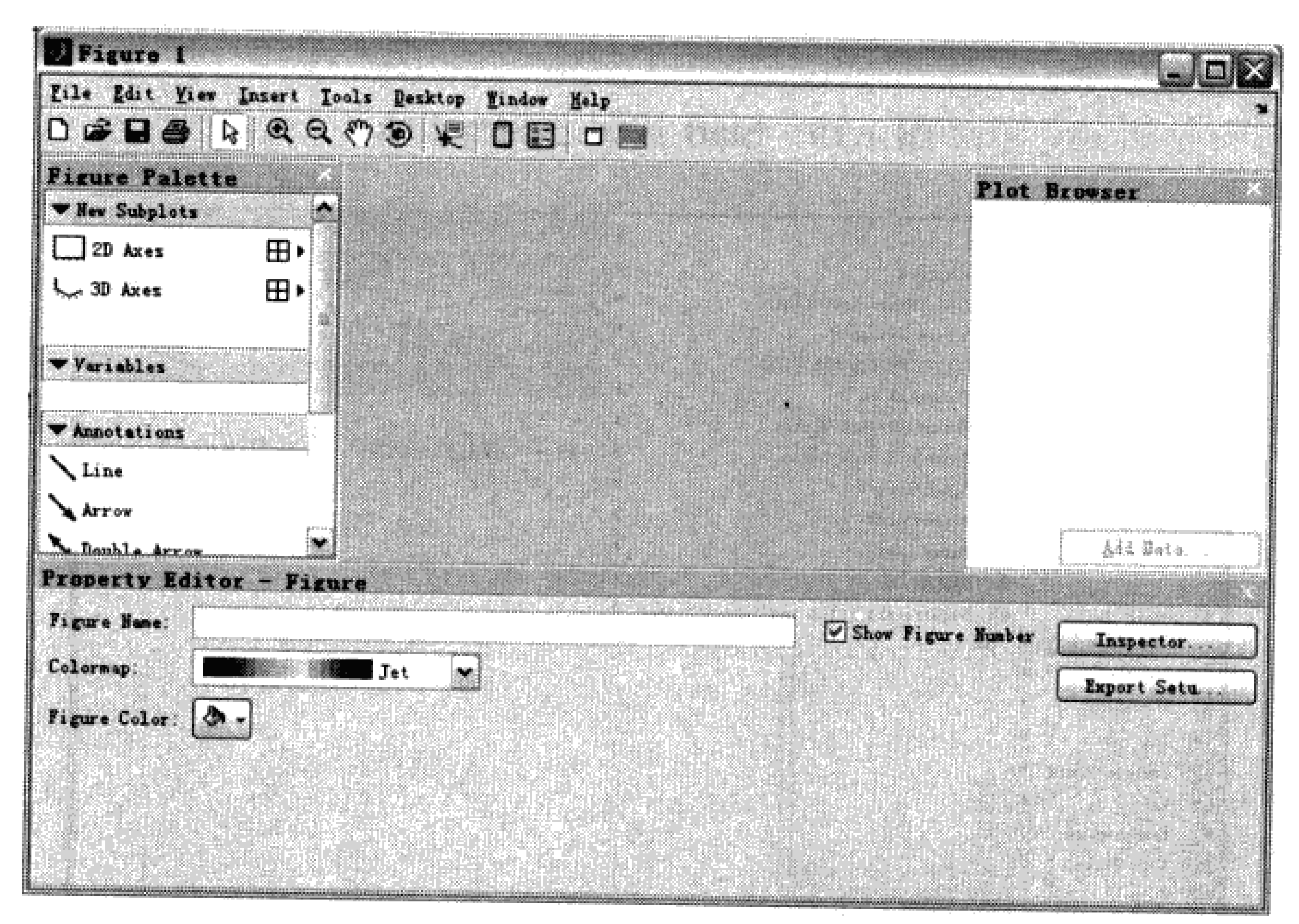


图 1.18　MATLAB 7.0 画图平台

1.3.8　MATLAB 的菜单栏

主窗口中除了嵌入的一些子窗口外，还包括菜单栏和工具栏。这

一小节主要介绍菜单栏。MATLAB 7.0 主窗口的菜单栏共包含 File、Edit、View、Graphics、Debug、Desktop、Window 和 Help 8 个菜单项。

1. File 菜单项

File 菜单项用于实现有关文件的操作，如图 1.19 所示。

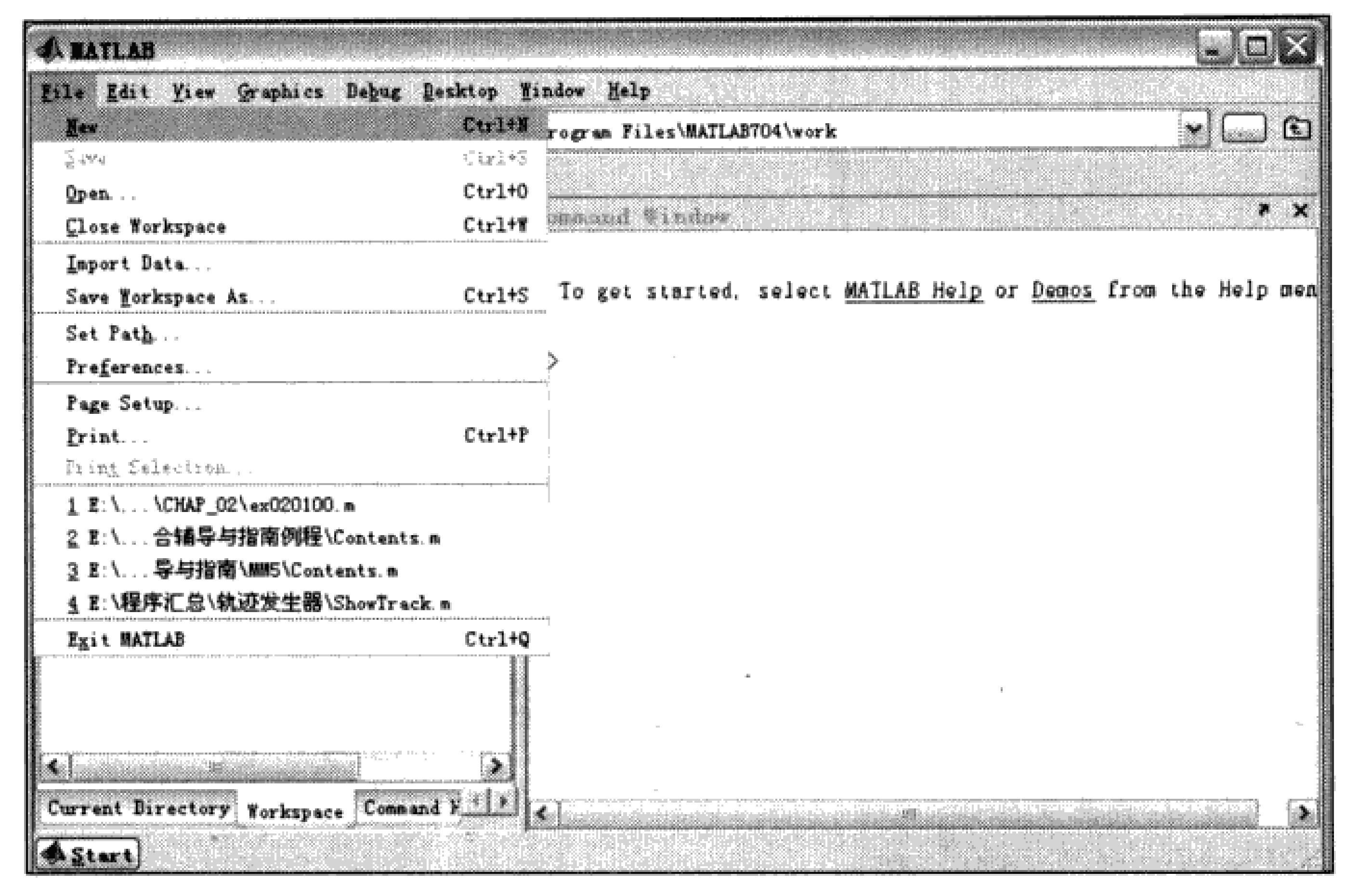


图 1.19 File 菜单项介绍

- New 命令：用于建立 M 文件、图形窗口。这里所说的 M 文件是指用 MATLAB 编写的程序文件。可以根据用户的需要改变默认名称“unnamed”为任何想要的名称，本例中取名“exem”，如图 1.20 所示。
- Open 命令：打开一个已经建立的 M 文件，如图 1.21 所示。
- Save Workspace As 命令：把当前工作空间的所有变量用扩展名为.mat 的文件保存起来，如图 1.22 所示，.mat 文件名称可以根据具体情况修改。
- Set Path 命令：打开 MATLAB 的路径浏览器。在如图 1.23 所示的 MATLAB 的路径浏览器中可以添加、移动文件夹。

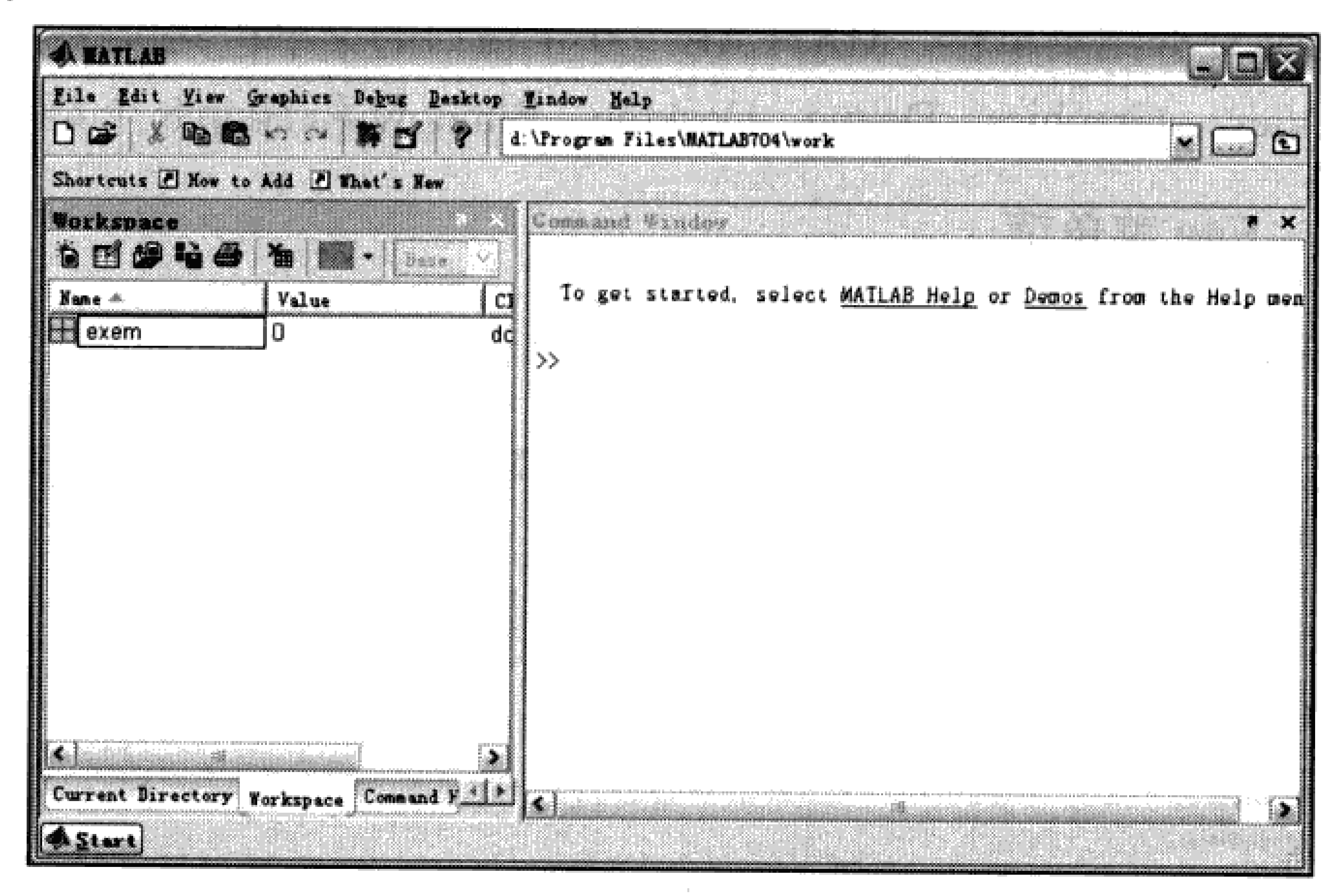


图 1.20　新建命令的使用

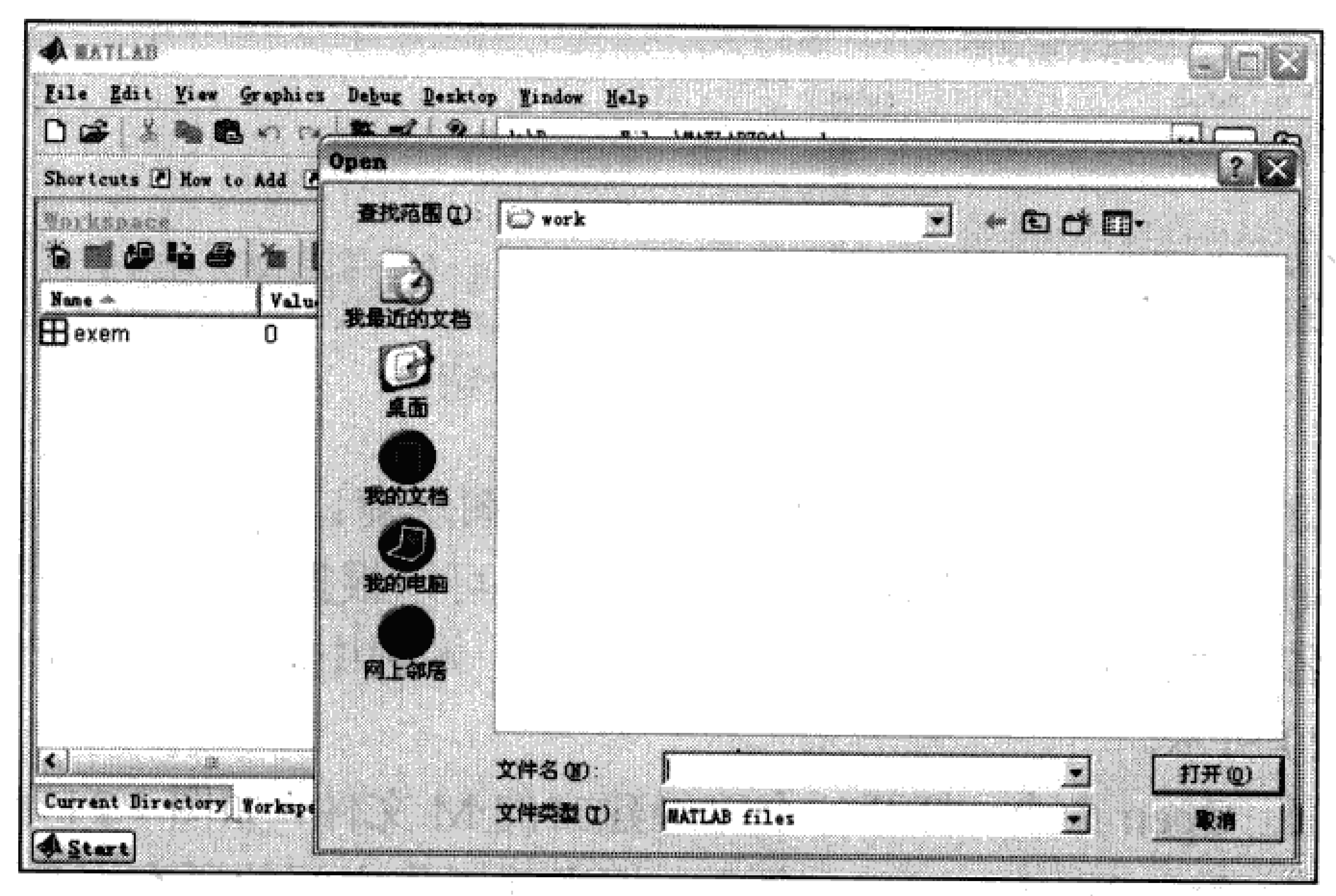


图 1.21　打开 MATLAB 文件

- Preferences 命令：打开如图 1.24 所示特征设置窗口进行相应的设置。
- Page Setup 命令：设置打印机的参数。

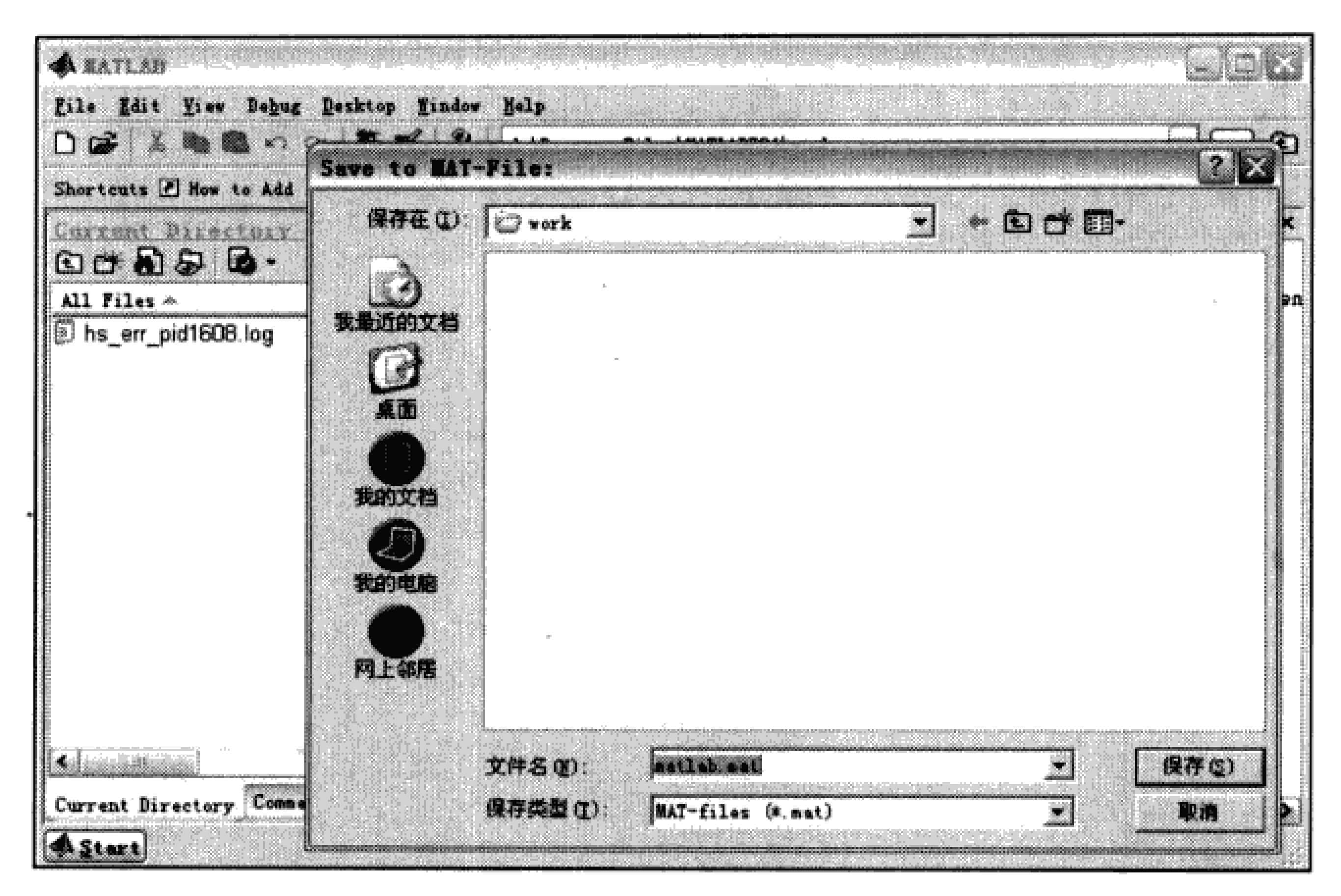


图 1.22 保存工作空间为.mat 文件

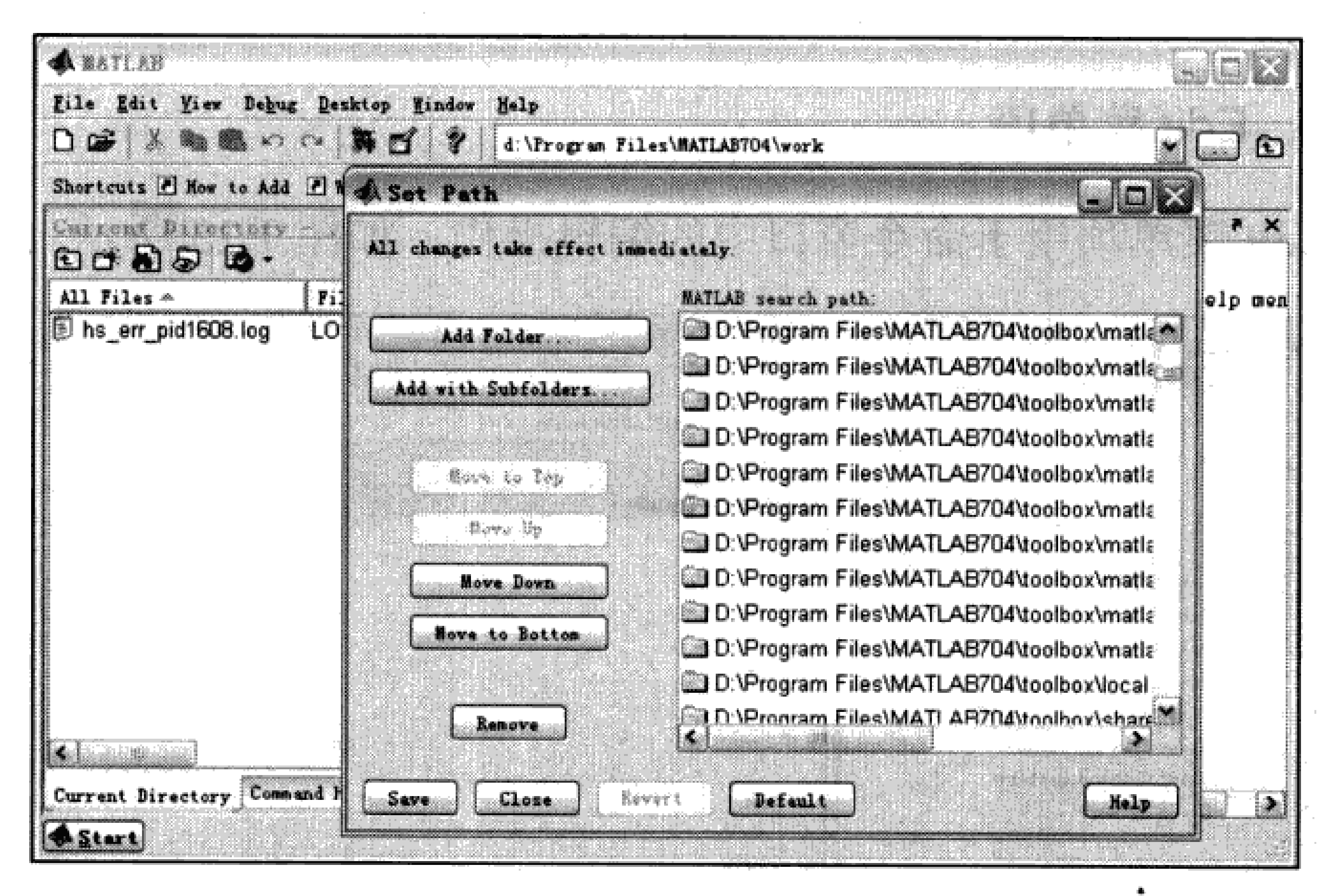


图 1.23 Set Path 设置窗口

- Print 命令：打印文件。
- Print Selection 命令：打印选中的内容。
- Exit MATLAB 命令：退出 MATLAB 系统。

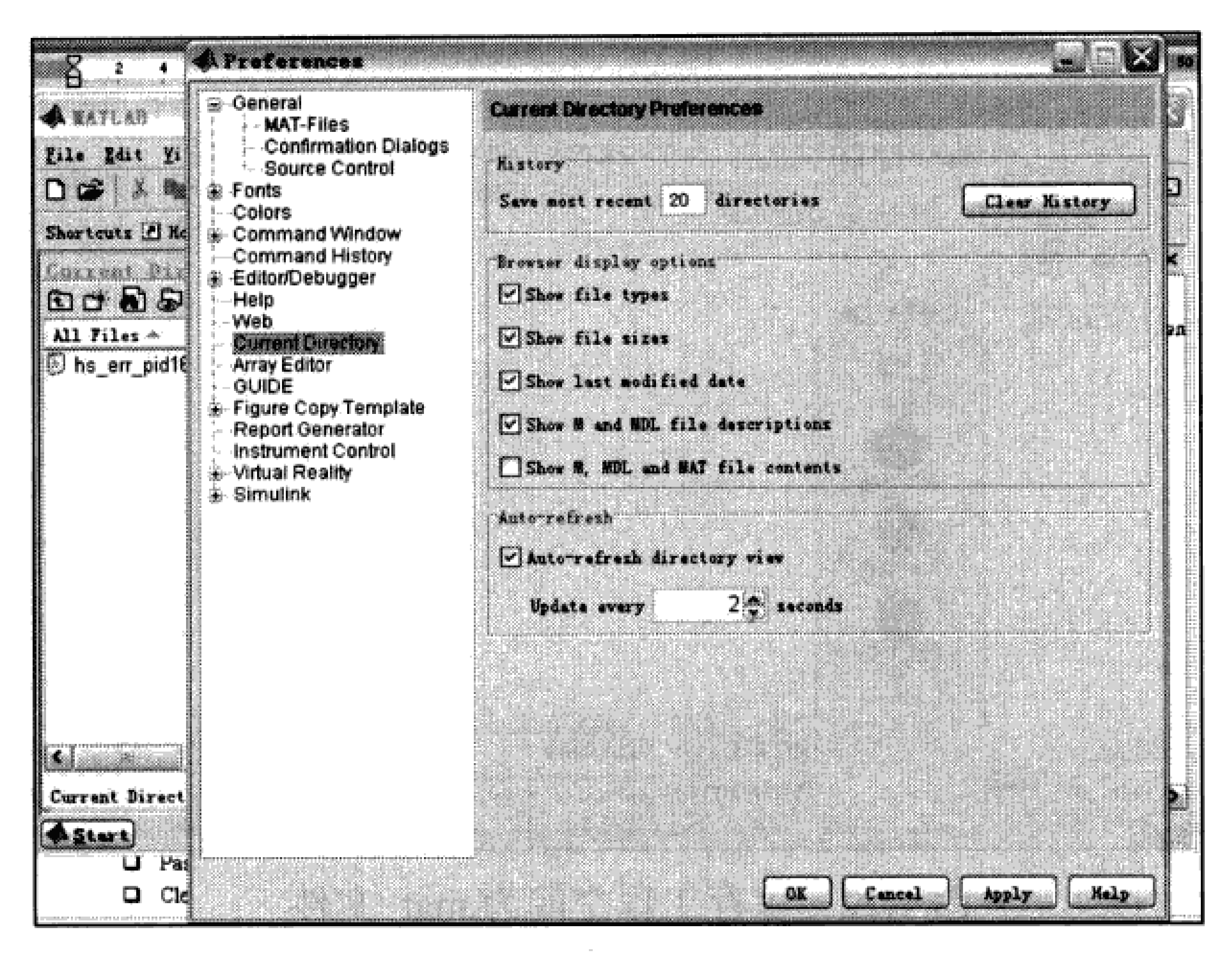


图 1.24 Preferences 设置窗口

2. Edit 菜单项

Edit 菜单项用于命令窗口中的编辑操作，如图 1.25 所示。

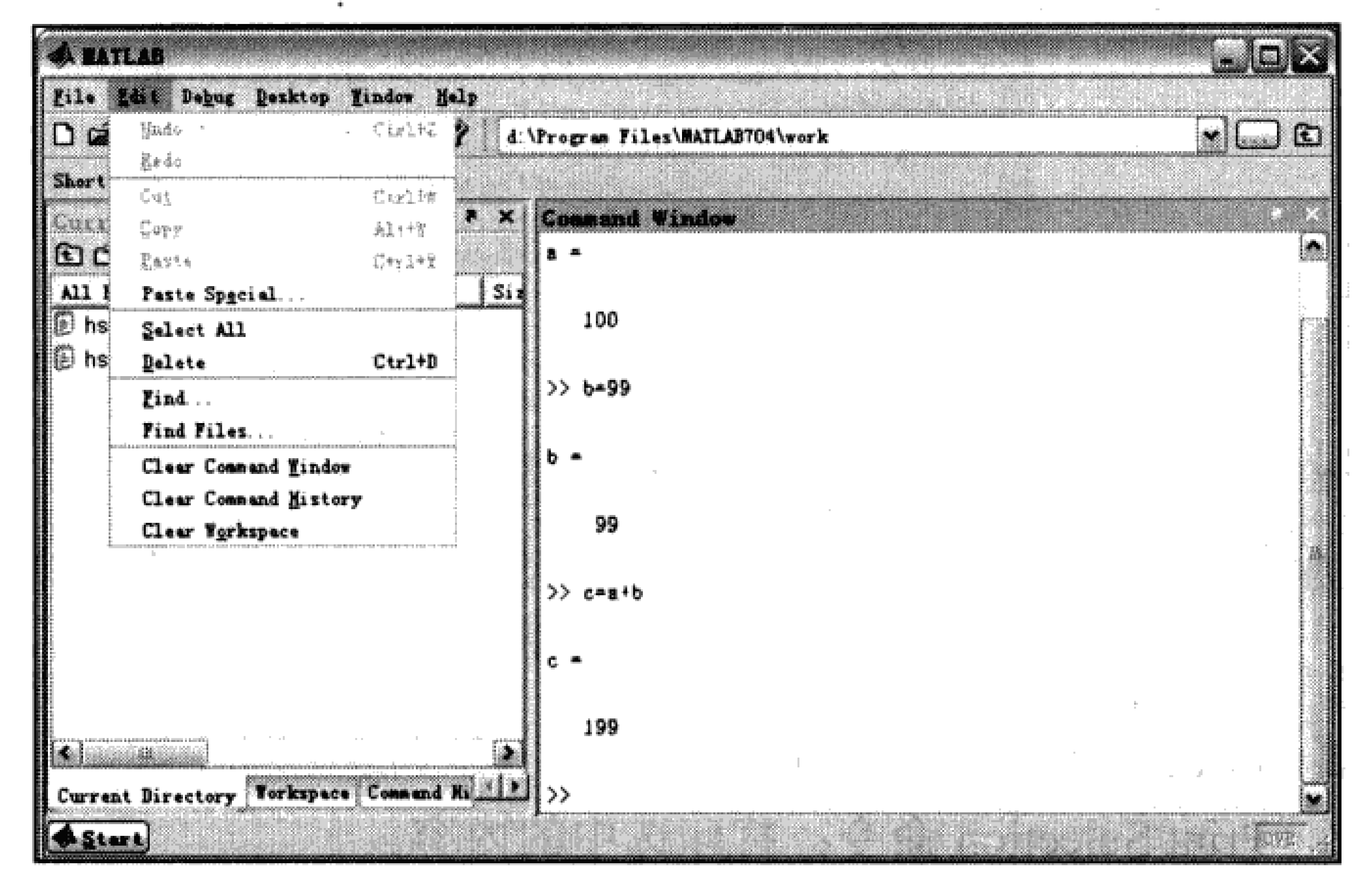


图 1.25 Edit 菜单项

- Undo 命令：撤销上一次操作。
- Cut 命令：剪切当前选定命令、程序或者图片。
- Copy 命令：复制当前所选命令、程序或者图片。
- Paste 命令：粘贴当前所选命令、程序或者图片。
- Delete 命令：删除内容。
- Select All 命令：用于选定所有文本内容。

以上为 Windows 中较为常用的命令，下面讲解几个 MATLAB 中特有的命令。

- Clear Command Window 命令：这个命令用来清除命令窗口中的内容，如图 1.26 所示为没有清除前的命令窗口，里面还有命令内容，而图 1.27 所示为执行"Clear Command Window"命令清除后的窗口。

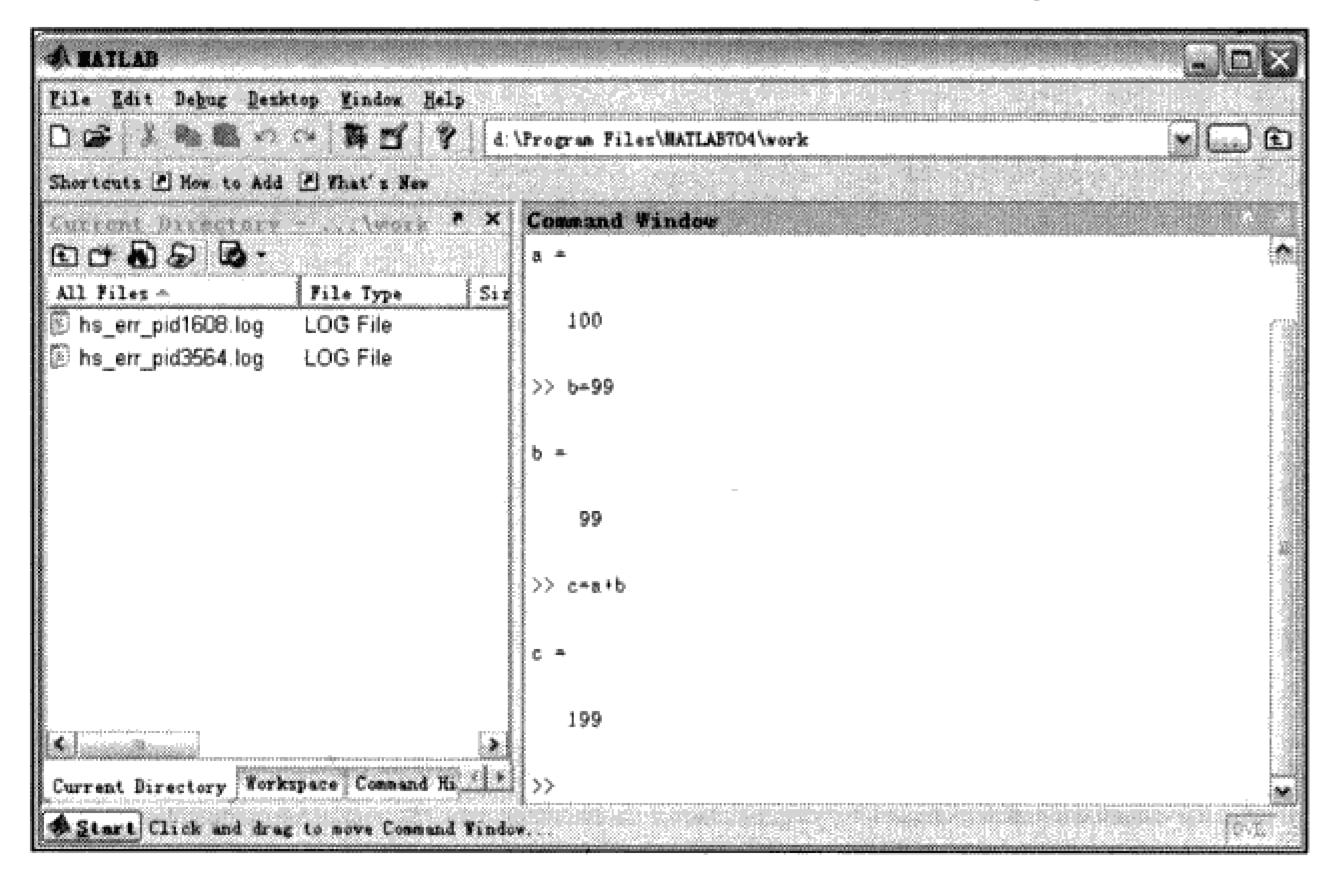


图 1.26 清除前的命令窗口

- Clear Command History 命令：清除命令历史。在 MATLAB 中每一次命令行或者 M 文件的操作，都会被 MATLAB 中的历史命令记录窗口记录，如果不需要这些记录，或者要重新记录新的历史，便会用到清除历史的命令。如图 1.28 所示为历史目录没有清除的窗口，图 1.29 所示为命令历史清除后的界面。

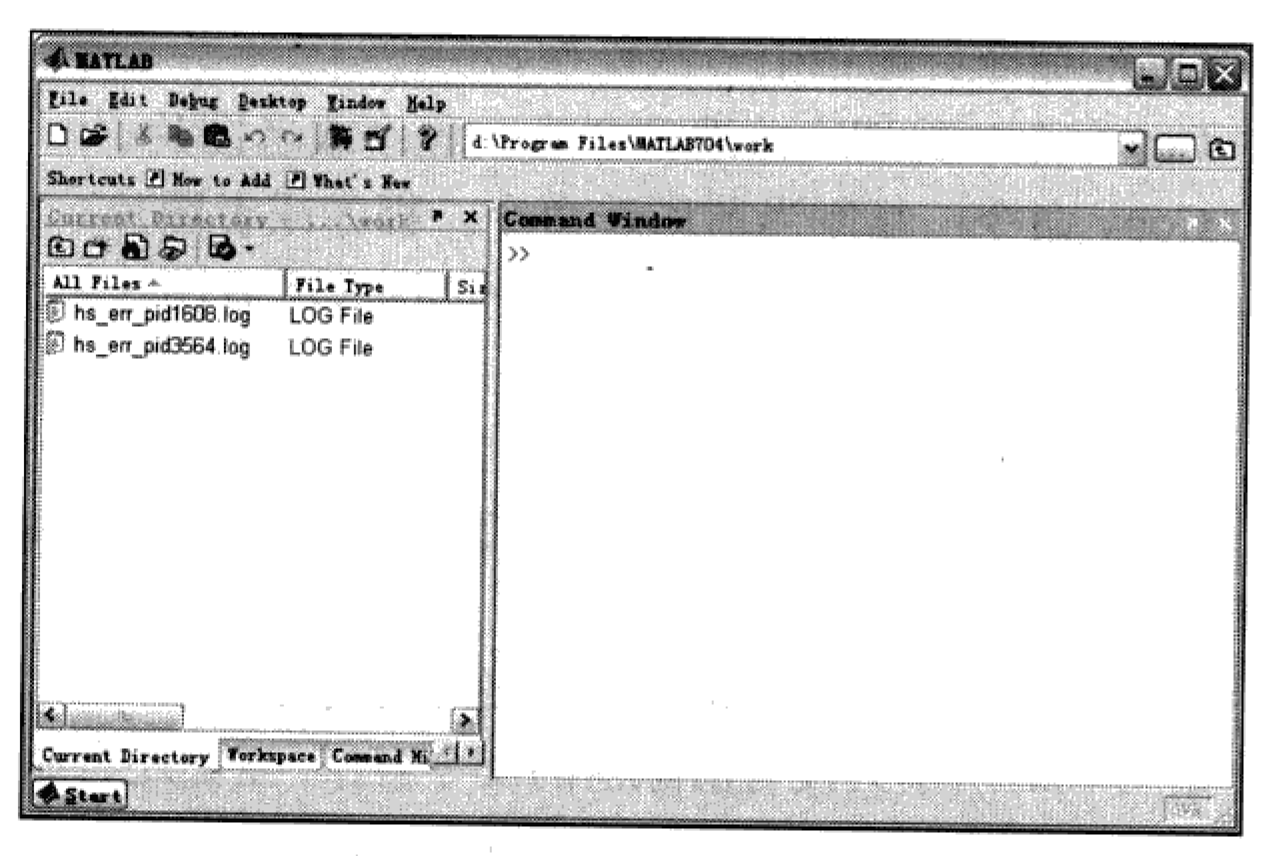


图 1.27　清除后的命令窗口

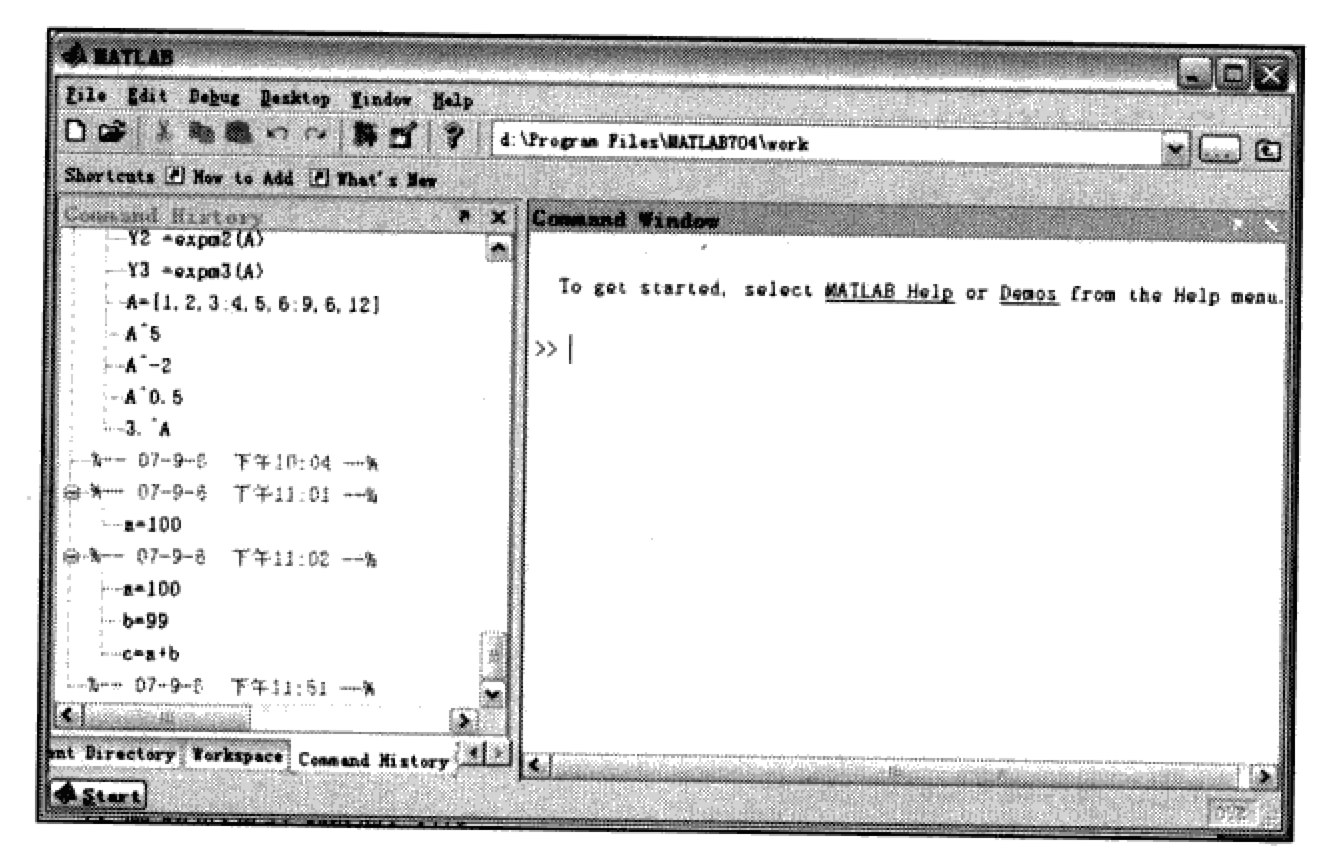


图 1.28　命令历史清除前

3. View 菜单项

View菜单项用于设置MATLAB集成环境的显示方式，如图1.30所示。

- Show Visual Directory 命令：显示可见的文件夹。如图 1.31所示左侧窗口中显示的内容即为当前文件夹下的路径，路径之间用

“|” 隔开。

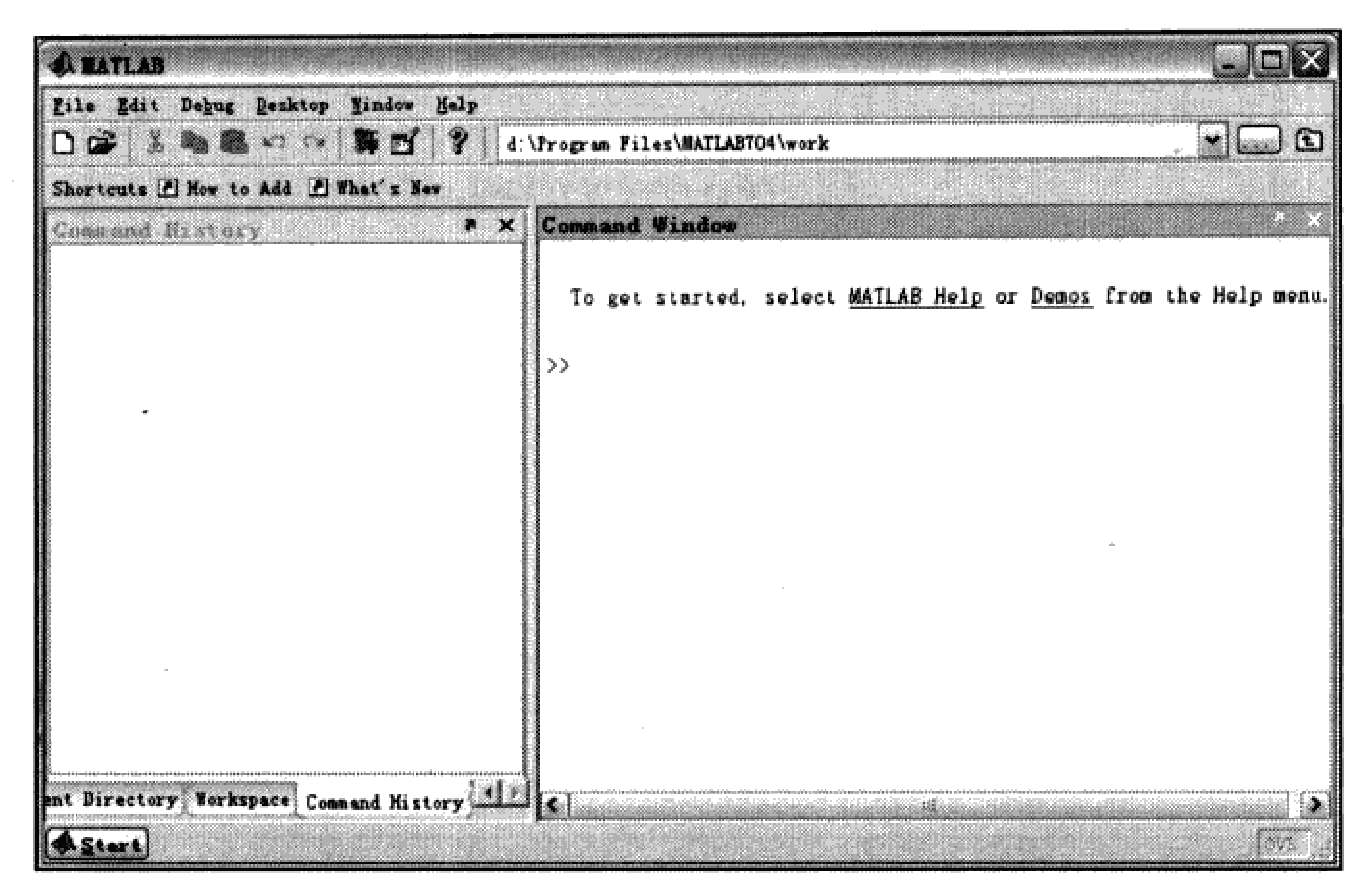


图 1.29 命令历史清除后

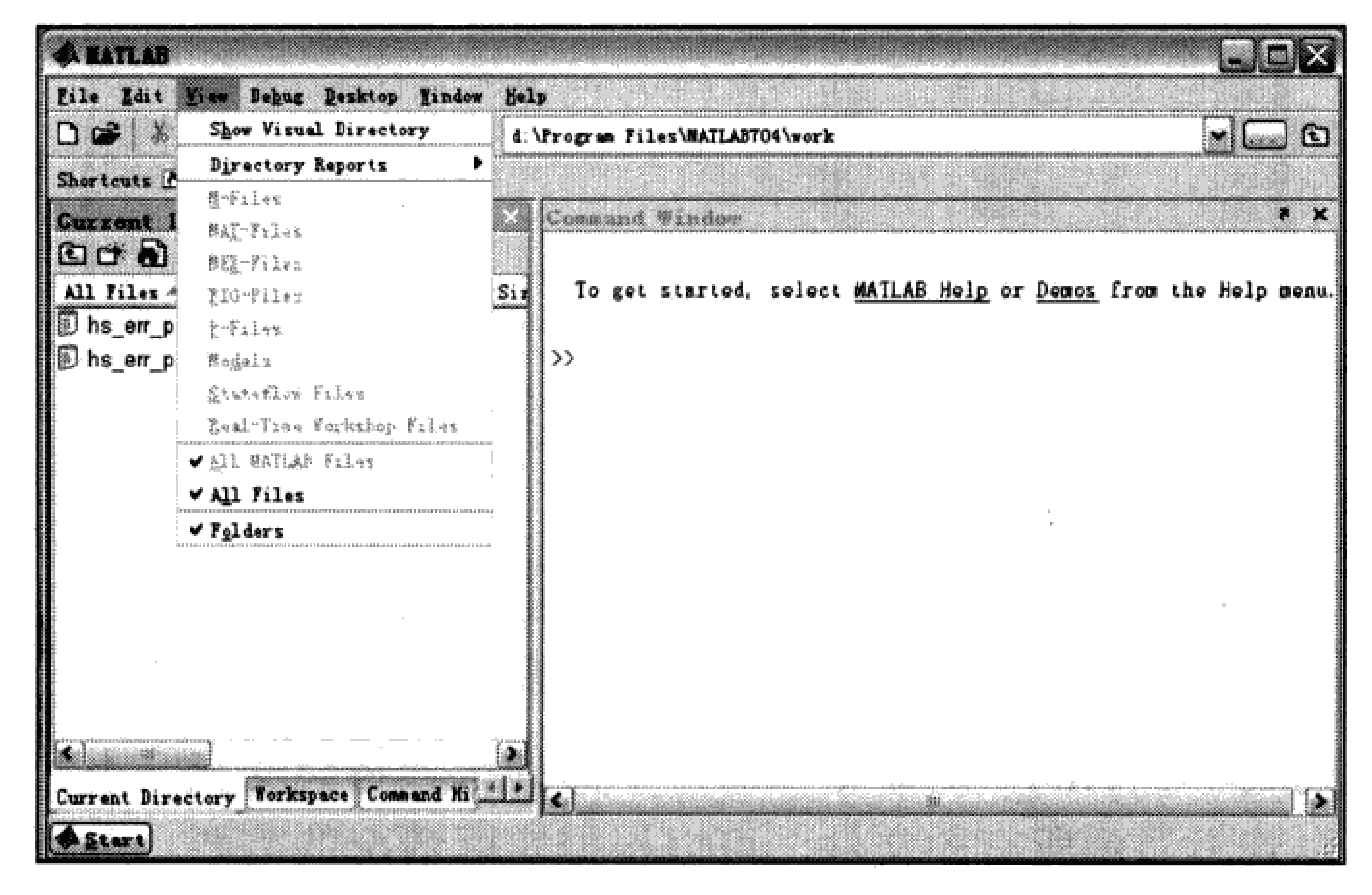


图 1.30 View 菜单项

■ All Files 命令：如果选中了该命令，表示将显示当前路径中所有类型的文件，显示结果如图 1.32 所示。

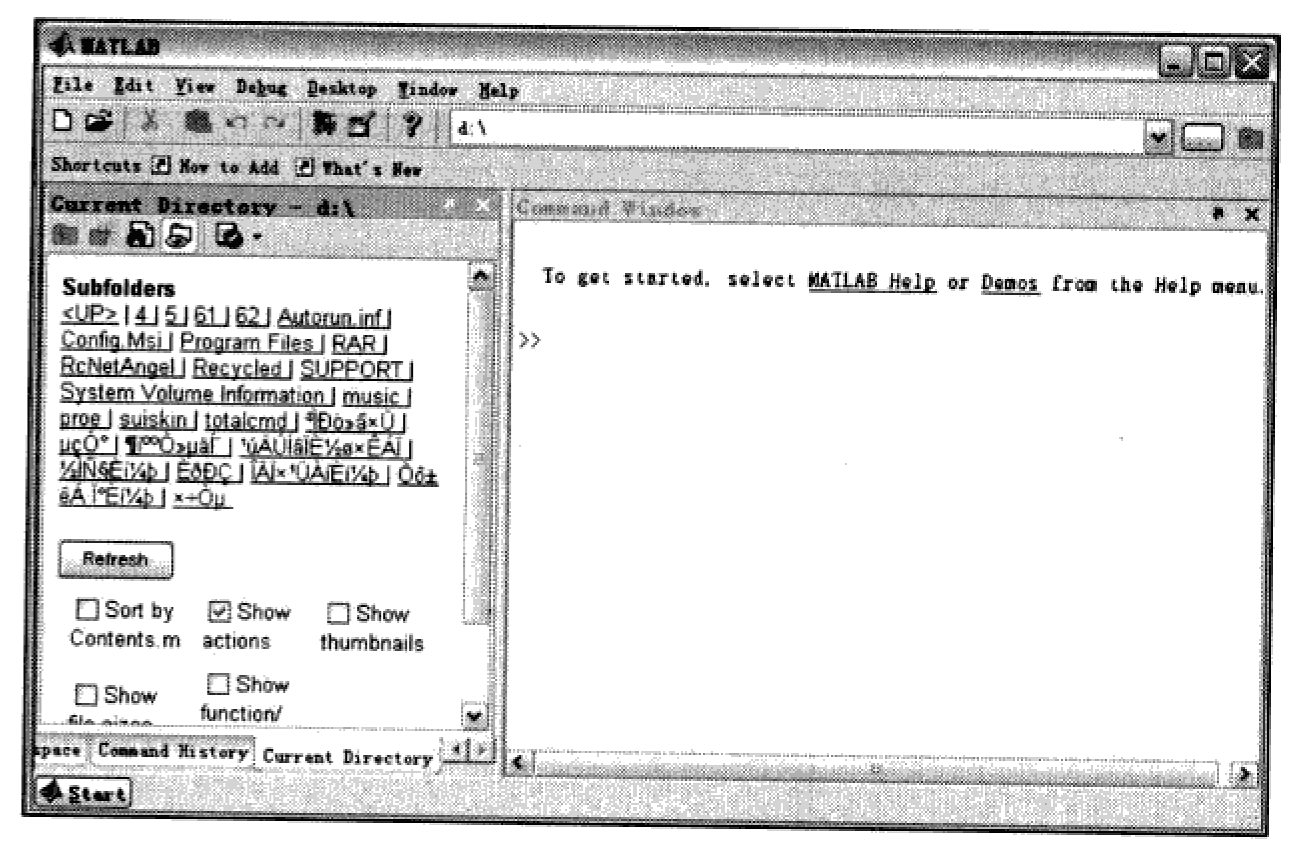


图 1.31　显示可见的路径

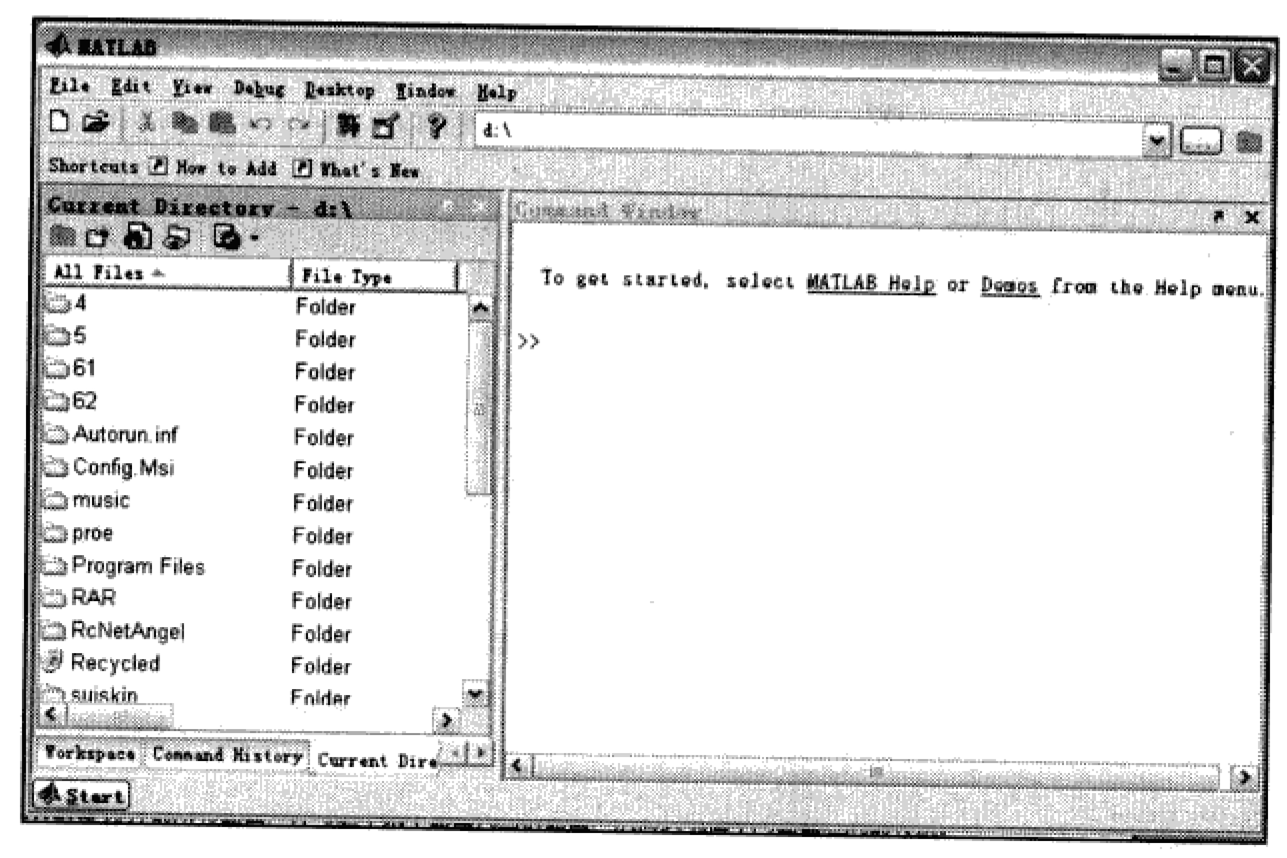


图 1.32　显示所有类型的文件

当选中了 All Files 命令项后，下面的选项将会变灰，不允许被选择。若不显示所有的文件，可以根据实际情况的需要，选择希望显示的文件类型。

- M-Files 命令：显示.m 文件。
- MAT-Files 命令：显示.mat 文件。

- MEX-Files 命令：显示.mex 文件。
- FIG-Files 命令：显示.fig 图形文件。
- P-Files 命令：显示.p 文件。
- Models 命令：显示模型文件。
- Real-Time Workshop Files 命令：实时显示工作状态文件。

4. Window 菜单项

主窗口菜单栏上的 Window 菜单中只包含一个子菜单 Close all，用于关闭所有打开的编辑器窗口，包括 M-file、Figure、Model 和 GUI 窗口。利用 Window 菜单项可以查看目前 MATLAB 打开的所有窗口，并可选中某个窗口为当前窗口，从而实现在不同窗口之间的转换。如图 1.33 所示。

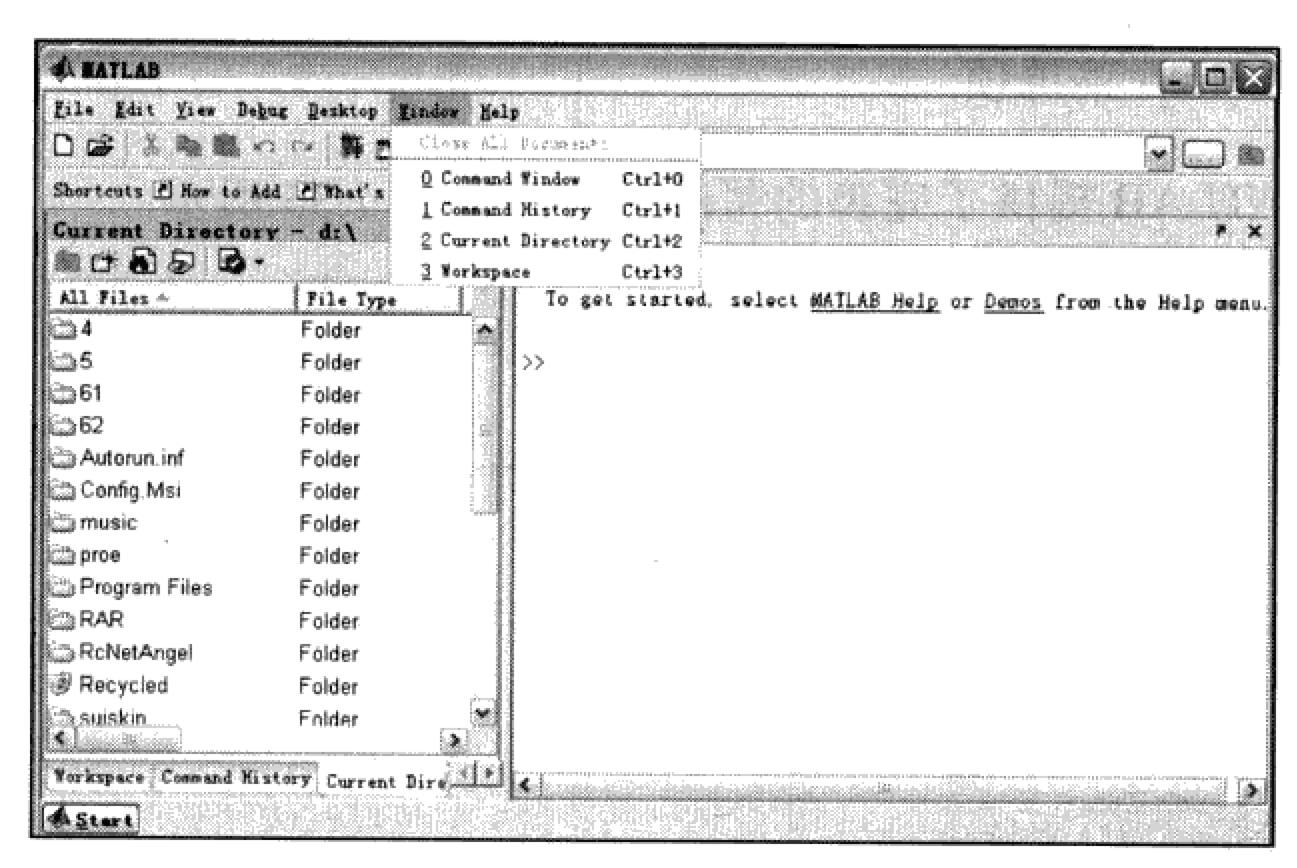


图 1.33 Window 菜单项

5. Desktop 菜单项

用于控制目前窗口中显示哪些子窗口，包括 MATLAB 的命令窗口、工作空间窗口、当前目录窗口、命令历史记录窗口等。如图 1.34 所示，打上标记的为当前显示的窗口，4 个窗口中，只有命

令窗口在界面的右侧，其余 3 个在界面的左侧。

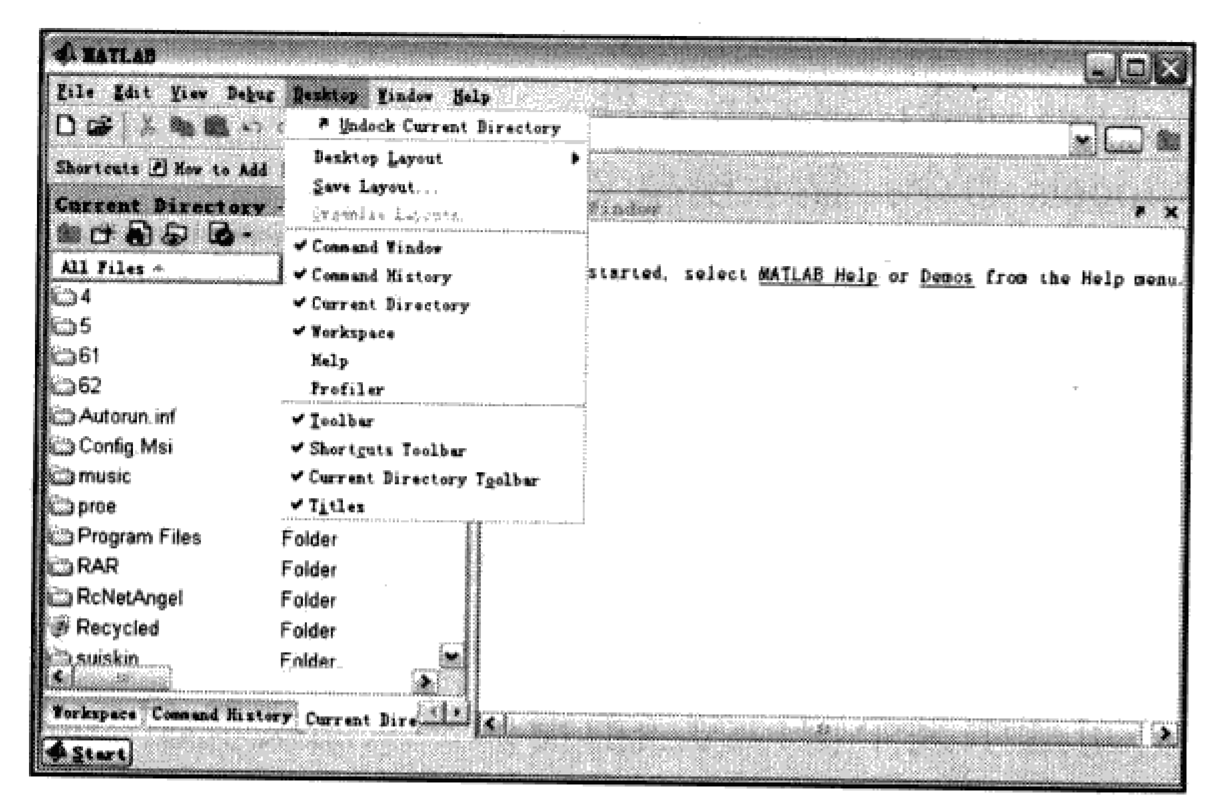


图 1.34　界面显示菜单项

6. Help 菜单项

MATLAB 提供了完善的帮助系统，如图 1.35 所示。

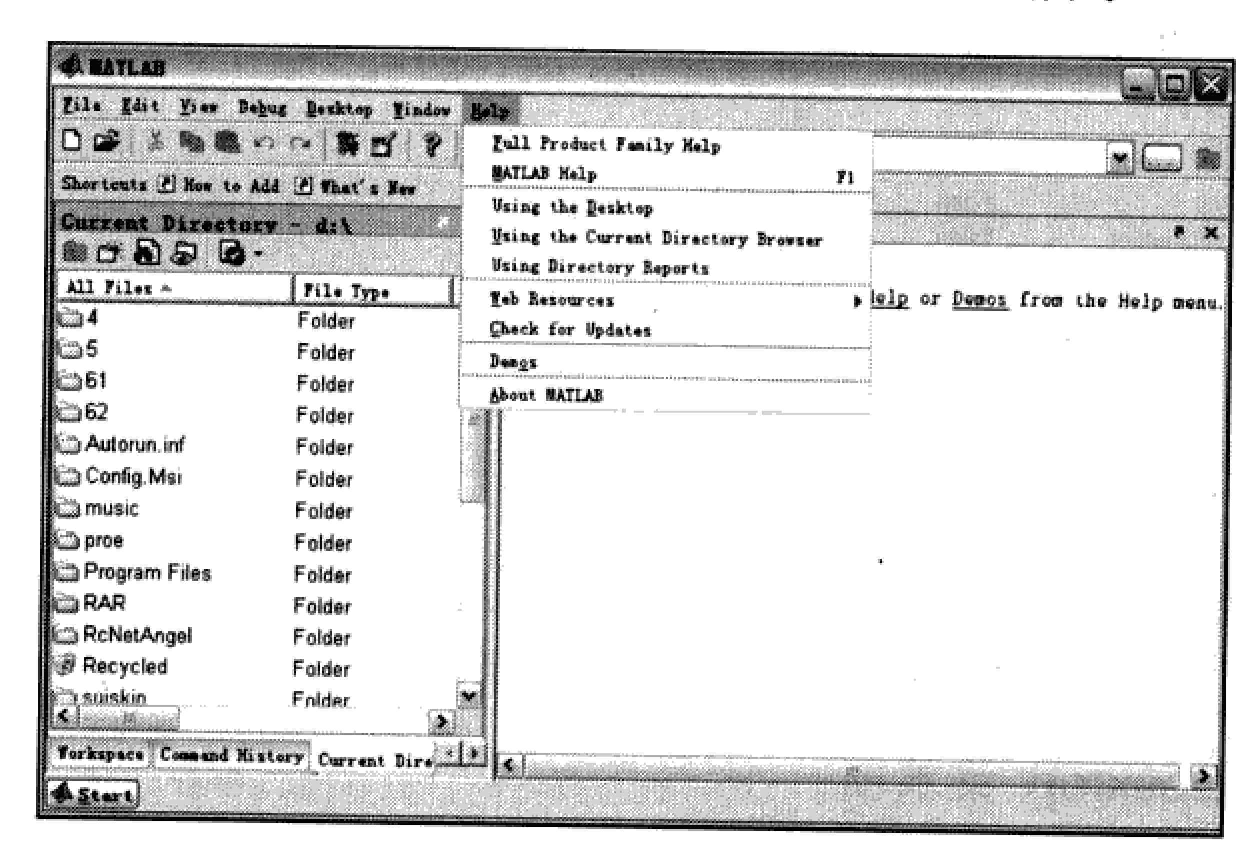


图 1.35　Help 菜单项

- Full Product Family Help 命令：打开 MATLAB 所有的产品帮助窗口，如图 1.36 所示。

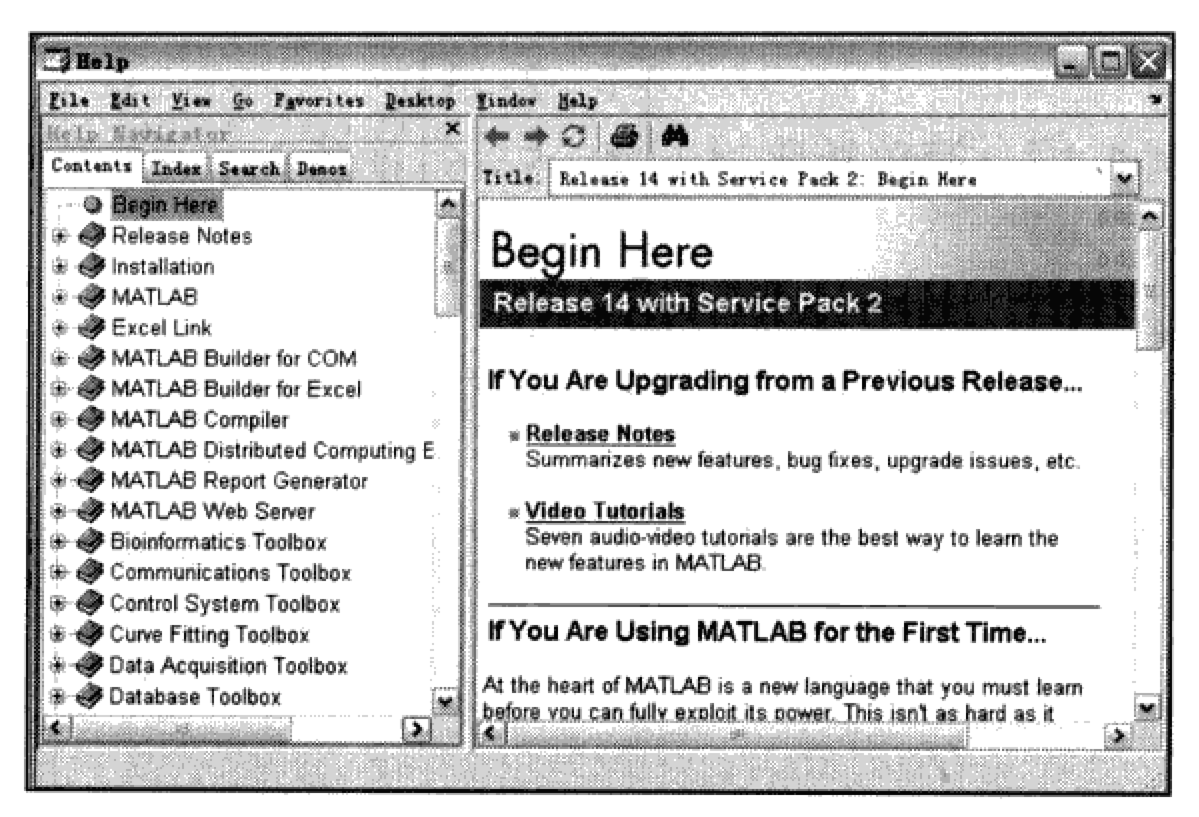


图 1.36 Full Product Family Help 命令窗口

1.3.9 MATLAB 的工具栏

MATLAB 7.0 主窗口中的工具栏共提供了 10 个命令按钮，如图 1.37 所示。

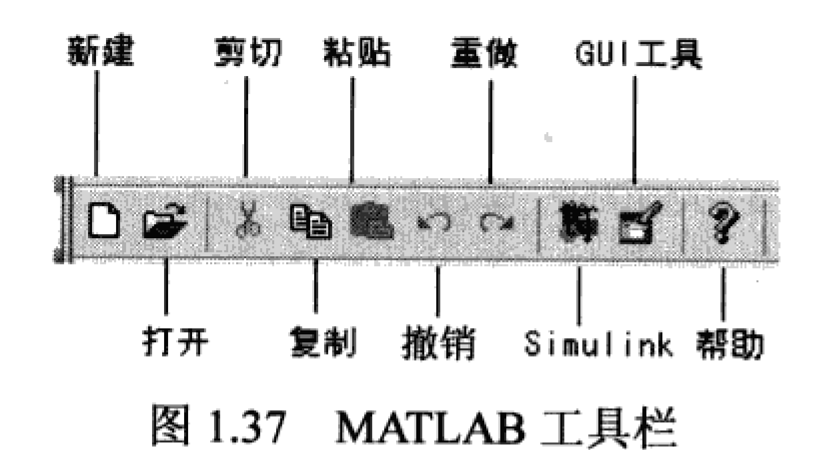


图 1.37 MATLAB 工具栏

1.4 MATLAB 入门实践

1.4.1 命令窗口操作

在本节中将要使用几个简单的运行实例讲解在命令窗口中操作

MATLAB 的方法。按照由简单到复杂的顺序循序渐地进行演示。

【实例 1.1】 简单矩阵 $A=\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\end{bmatrix}$ 的输入步骤。

（1）在命令窗口中输入下列内容。

```
>> A = [1,2,3; 4,5,6; 7,8,9]
```

按【Enter】键，指令被执行。

（2）在指令执行后，MATLAB 命令窗口中将显示以下结果：

```
A =
     1     2     3
     4     5     6
     7     8     9
```

【实例讲解】 从上面的结果中可以看出，在 MATLAB 中，逗号表示同行，而分号表示换行。

【实例 1.2】 复数矩阵的生成及运算。

```
>>A=[1,3;2,4]-[5,8;6,9]*i
A =
   1.0000 - 5.0000i   3.0000 - 8.0000i
   2.0000 - 6.0000i   4.0000 - 9.0000i
>>B=[1+5i,2+6i;3+8*i,4+9*i]
B =
   1.0000 + 5.0000i   2.0000 + 6.0000i
   3.0000 + 8.0000i   4.0000 + 9.0000i
>>C=A*B
C =
  1.0e+002 *
   0.9900             1.1600 - 0.0900i
   1.1600 + 0.0900i   1.3700
```

【实例讲解】 从上面的代码中可以看出，在 MATLAB 中处理复数的时候，只需要将实部和虚部分开就可以了，十分便利。

【实例 1.3】 求实例 1.2 中复数矩阵 C 的实部、虚部、模和相角。

```
C_real=real(C)
C_imag=imag(C)
C_magnitude=abs(C)
```

```
C_phase=angle(C)*180/pi                    %以度为单位计算相角
C_real =
    99   116
   116   137
C_imag =
     0    -9
     9     0
C_magnitude =
   99.0000  116.3486
  116.3486  137.0000
C_phase =
         0   -4.4365
4.4365         0
```

【实例讲解】 在 MATLAB 中，将复数的实部和虚部当作不同的数组对象来处理。

1.4.2 计算结果的图形表示

【实例 1.4】 画出衰减振荡曲线 $y = e^{-\frac{t}{3}}\sin 3t$ 及其他的包络线 $y_0 = e^{\frac{t}{3}}$。t 的取值范围是$[0,4\pi]$。

```
t=0:pi/50:4*pi;                        %定义自变量取值数组
y0=exp(-t/3);                          %计算与自变量相应的 y0 数组
y=exp(-t/3).*sin(3*t);                 %计算与自变量相应的 y 数组
plot(t,y,'-r',t,y0,':b',t,-y0,':b')    %用不同颜色、线型
绘制曲线
grid                                   %在“坐标纸”画小方格
```

通过上面指令运行后得到的图形如图 1.38 所示。

【实例讲解】 在 MATLAB 中用 pi 来表示圆周率，函数 exp（x）表示数学计算中的e^x，plot 为 MATLAB 中的绘图命令，grid 为画小方格命令，以上的所有命令都会在后面各章作详细的讲解。

【实例 1.5】 画出 $z = \dfrac{\sin(\sqrt{x^2+y^2})}{\sqrt{x^2+y^2}}$ 所表示的三维曲面。x, y 的取值范围是$[-8,8]$。

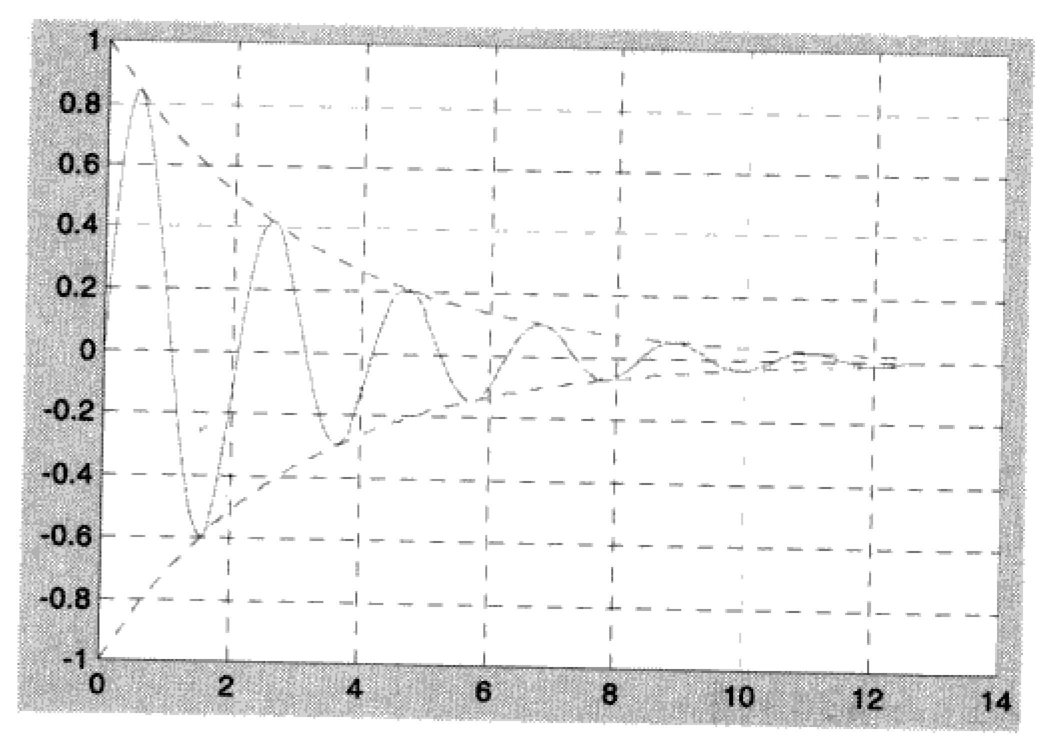


图 1.38　衰减振荡曲线与包络

```
clear;
x=-8:0.5:8;                    %定义自变量 x 的一维刻度向量
y=x';                          %定义自变量 y 的一维刻度向量
X=ones(size(y))*x;%计算自变量平面上取值点 x 坐标的二维数组
Y=y*ones(size(x));%计算自变量平面上取值点 y 坐标的二维数组
R=sqrt(X.^2+Y.^2)+eps;%计算中间变量 R=sqrt(x^2+y^2)    <5>
Z=sin(R)./R; %计算与自变量二维数组相应的函数值 z=sinR/R   <6>
mesh(Z);                              %绘制三维网格图
colormap(hot)                         %指定网格图用 hot 色图绘制
```

上例运行后得到的图形如图 1.39 所示。

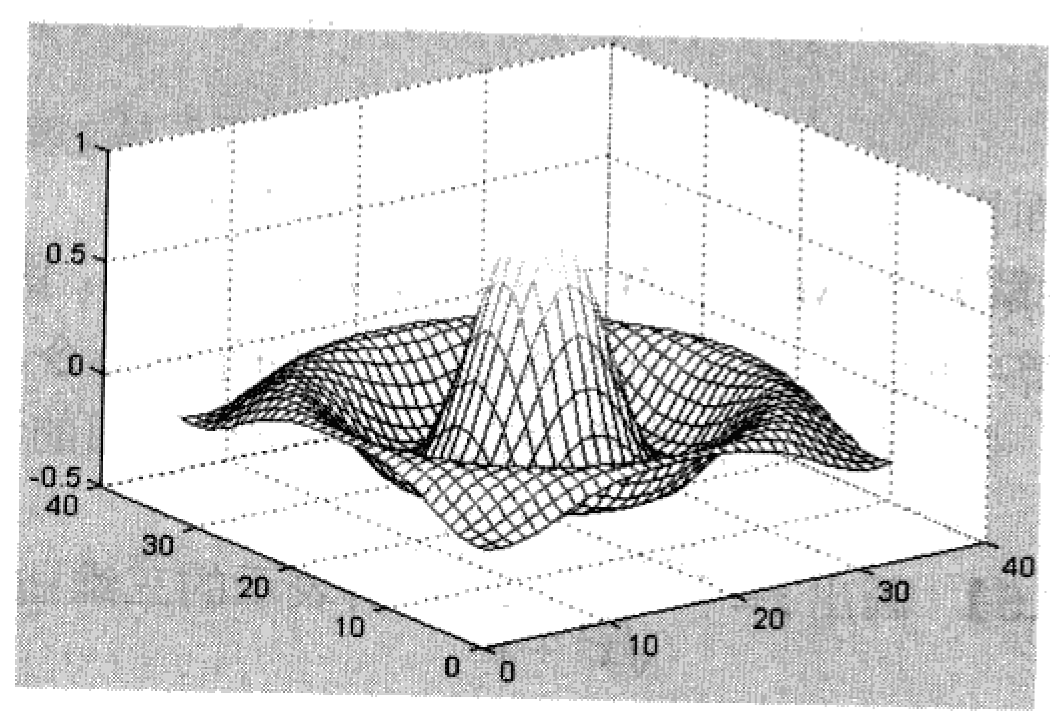


图 1.39　三维网格图绘制结果

【实例讲解】 sqrt表示数值计算中的开平方根，sin即为数学中的正弦函数，mesh用来绘制三维网格图，colormap命令用来指定色图的绘制形式，本例中指定用hot色图绘制。

1.4.3 内存变量的查阅命令——who或whos

【实例1.6】 用who检查MATLAB内存变量。

```
who
Your variables are:

R          Y          x          y1
X          Z          y          y2
```

【实例讲解】 运行以上命令后就可看到内存变量的信息。

【实例1.7】 输入whos，获得驻留变量的详细情况：全部变量名、变量的数组维数、占用字节数、变量的类别（如双精度）、是否复数等。

```
whos
  Name       Size            Bytes  Class
  R         33x33            8712   double array
  X         33x33            8712   double array
  Y         33x33            8712   double array
  Z         33x33            8712   double array
  x          1x33            264    double array
  y         33x1             264    double array
  y1         1x1             8      double array
  y2         1x1             8      double array
Grand total is 4424 elements using 35392 bytes
```

【实例讲解】 在比较复杂的程序开发中，常需要通过whos命令来查看变量信息。

1.4.4 变量的文件保存命令——save和load命令

【实例1.8】 数据的存取。

```
mkdir('c:\','my_dir');                  %在C盘上创建目录my_dir
cd c:\my_dir     %使c:\my_dir成为当前目录
```

```
save  saf X Y Z %选择内存中的X、Y、Z变量保存为saf.mat文件
dir              %显示目录上的文件
.     ..    saf.mat

%清空内存,从saf.mat 向内存装载变量Z
clear            %清除内存中的全部变量
load  saf  Z     %把saf.mat文件中的Z变量装入内存
who              %检查内存中有什么变量
Your variables are:
Z
```

【实例讲解】 如果一组数据是经过长时间的复杂计算后获得的，那么为避免再次重复计算，常使用 save 命令加以保存。此后，每当需要，都可通过 load 命令重新获取这组数据。

1.5 MATLAB 帮助系统

1.5.1 帮助窗口

进入帮助窗口可以通过以下 3 种方法：

- 单击 MATLAB 主窗口工具栏中的 Help 按钮；
- 在命令窗口中输入 helpwin、helpdesk 或 doc；
- 选择 Help 菜单中的“MATLAB Help”命令。

1.5.2 帮助命令

MATLAB 帮助命令包括 help、lookfor 以及模糊查询。在 MATLAB 命令窗口中直接输入 help 命令将会显示当前帮助系统中所包含的所有项目，即搜索路径中所有的目录名称。同样，可以通过“help+函数名”的形式来显示该函数的帮助说明。

（1）直接使用 help 获得命令的使用说明。

【功能介绍】 假如准确知道所要求助的主题词或命令名称，那

么使用 help 是获得在线帮助的最简单有效的途径。

【实例 1.9】 关于矩阵对数函数 logm 使用说明的在线求助。

```
>> help logm
 LOGM   Matrix logarithm.
L = LOGM(A) is the matrix logarithm of A, the inverse
of EXPM(A). Complex results are produced if A has negative
eigenvalues.   A warning message is printed if the computed
expm(L) is not close  to A.

[L,esterr] = logm(A) does not print any warning message,
but returns an estimate of the relative residual, norm(expm
(L)-A)/norm(A).

If A is real symmetric or complex Hermitian, then so is
LOGM(A).

Some matrices, like A = [0 1; 0 0], do not have any
logarithms, real or complex, and LOGM cannot be expected to
produce one.

See also EXPM, SQRTM, FUNM.
```

（2）使用 help 命令进行分类搜索。

【实例 1.10】 运行不带任何限定的 help，可以得到分类名称明细表。

```
>> help
HELP topics:
matlab\general   -  General purpose commands.
matlab\ops       -  Operators and special characters.
matlab\lang      -  Programming language constructs.
matlab\elmat     -   Elementary matrices and matrix
manipulation.
matlab\elfun     -  Elementary math functions.
matlab\specfun   -  Specialized math functions.
......              ......
For more help on directory/topic, type "help topic".
```

（3）使用 help topic 命令形式获得具体子类的命令明细。

【实例 1.11】 如果用户想知道有关矩阵操作的所有命令，那么通过运行以下命令就可以实现。

```
>> help elmat
  Elementary matrices and matrix manipulation.
  Elementary matrices.
    zeros        - Zeros array.
    ones         - Ones array.
    ......         ......
  Basic array information.
    size         - Size of matrix.
    length       - Length of vector.
    ......         ......
  Matrix manipulation.
    reshape      - Change size.
    diag     - Diagonal matrices and diagonals of matrix.
    ......         ......
  Special variables and constants.
    ans          - Most recent answer.
    eps          - Floating point relative accuracy.
    ......         ......
  Specialized matrices.
    compan       - Companion matrix.
    gallery      - Higham test matrices.
```

（4）lookfor 命令。

help 命令只能搜索出那些关键字完全匹配的结果，lookfor 命令对搜索范围内的 M 文件进行关键字搜索，条件比较宽松。lookfor 命令只对 M 文件的第一行进行关键字搜索。若在 lookfor 命令加上 -all 选项，则可对 M 文件进行全文搜索。

【实例 1.12】 查找包含“integral”这个关键词的所有命令。

```
>> lookfor integral
ELLIPKE      Complete elliptic integral.
EXPINT       Exponential integral function.
DBLQUAD      Numerically evaluate double integral.
INNERLP      Used with DBLQUAD to evaluate inner loop of
integral.
```

```
    QUAD           Numerically evaluate integral, low order
method.
    QUAD8          Numerically evaluate integral, higher order
method.
    COSINT         Cosine integral function.
    SININT         Sine integral function.
    ASSEMA              Assembles area integral contributions
in a PDE problem.
    COSINT         Cosine integral function.
    FOURIER        Fourier integral transform.
    IFOURIER       Inverse Fourier integral transform.
    SININT         Sine integral function.
```

（5）模糊查询。

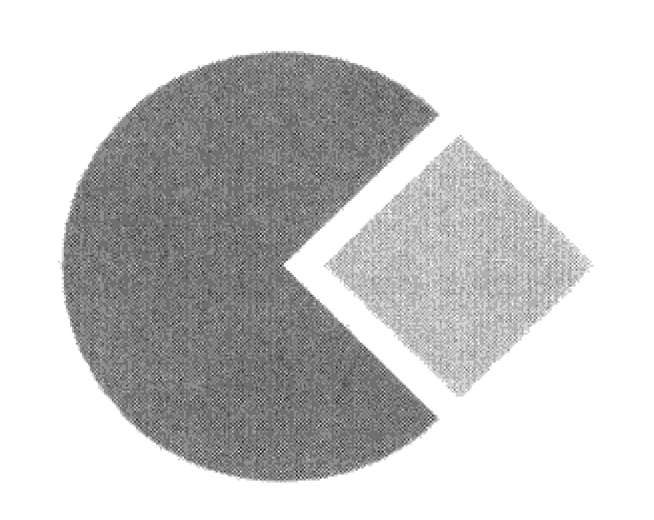
MATLAB 6.0 以上的版本提供了一种类似模糊查询的命令查询方法，用户只需要输入命令的前几个字母，然后按 Tab 键，系统就会列出所有以这几个字母开头的命令。

1.5.3 演示系统

在帮助窗口中选择演示系统（Demos）选项卡，然后在其中选择相应的演示模块，或者在命令窗口输入 Demos，或者选择主窗口 Help 菜单中的 Demos 子菜单，都可以打开演示系统。

1.5.4 远程帮助系统

在 MathWorks 公司的主页（http://www.mathworks.com）上可以找到很多有用的信息，国内的一些网站也有丰富的信息资源。



第 2 章 矩阵及其基本运算

MATLAB，也称为“矩阵实验室”，它是以矩阵为基本运算单元的。因此，本书从最基本的运算单元出发，介绍 MATLAB 的函数及其用法。

2.1 矩阵的表示

MATLAB 的强大功能之一体现在能直接处理向量或矩阵。当然首要任务是输入待处理的向量或矩阵。矩阵是 MATLAB 学习的基本单元，那么对于矩阵本身的学习而言，创建矩阵则是学习矩阵及其基本运算的开端。本小节将主要讲述在 MATLAB 系统中如何生成满足学习和工程需要的各种矩阵。

2.1.1 实数矩阵输入

【语法说明】 命令行直接输入法。

【功能介绍】 通过命令行直接输入实数值，用分号（;）控制行数，建立实数矩阵。

【实例 2.1】 利用命令行直接输入法建立实数矩阵。

```
>> Time = [1  2  3  4  5  6  7  8  9  10  11  12  13]
Time =
```

```
1  2  3  4  5  6  7  8  9  10  11  12  13
>> X_Da = [2.32  3.43;4.37  5.98]
X_Da =
2.43  3.43
4.37  5.98
>> vect_a = [1  2  3  4  5]
vect_a =
1  2  3  4  5
>> Matrix_B = [1  2  3;
>>              2  3  4;3  4  5]
Matrix_B = 1  2  3
2  3  4
3  4  5
>> Null_M = [ ]            %生成一个空矩阵
```

【实例讲解】 无论任何矩阵（向量），都可以直接按行方式输入每个元素：同一行中的元素用逗号（,）或者用空格符来分隔，且空格个数不限；不同的行用分号（;）分隔。所有元素处于一方括号（[]）内；当矩阵是多维（三维以上），且方括号内的元素是维数较低的矩阵时，会有多重的方括号。

2.1.2　复数矩阵输入

【语法说明】 命令行直接输入法。

【功能介绍】 通过命令行直接输入数值，建立复数矩阵。

【实例 2.2】 利用命令行直接输入法建立复数矩阵，方式之一。

```
>> a=2.7;b=13/25;
>> C=[1,2*a+i*b,b*sqrt(a); sin(pi/4),a+5*b,3.5+1]
C=
   1.0000              5.4000 + 0.5200i   0.8544
   0.7071              5.3000             4.5000
```

用第二种方式建立复数矩阵：

```
>> R=[1 2 3;4 5 6], M=[11 12 13;14 15 16]
R =
```

```
  1    2    3
  4    5    6
M =
  11    12    13
  14    15    16
>> CN=R+i*M
CN =
   1.0000+11.0000i   2.0000+12.0000i   3.0000 +13.0000i
   4.0000+14.0000i   5.0000+15.0000i   6.0000 +16.0000i
```

【实例讲解】 两种输入方式稍有不同，第一种方式先输入数值，再代入矩阵；第二种方式为先输入矩阵，再对矩阵做相应操作。

2.1.3 sym 函数——定义符号矩阵

在 MATLAB 中输入符号向量（矩阵）的方法和输入数值类型的向量（矩阵）在形式上很相像，只不过要用到符号矩阵定义函数 sym，或者是用到符号定义函数 syms，先定义一些必要的符号变量，再像定义普通矩阵一样输入符号矩阵。

【语法说明】 sym（符号量 1；符号量 2；符号量 n）。

【功能介绍】 定义符号矩阵。

【实例 2.3】 用 sym 函数定义符号矩阵。

```
>>sym_matrix = sym('[A B C;NIHAO,Welcome,to beijing],')
sym_matrix =
[A          B          C]
[NIHAO    Welcome    to beijing]
>> sym_digits = sym('[1 2 3;a b c;sin(x) cos(y) tan
(z)]')
sym_digits =
[1          2          3]
[a          b          c]
[sin(x) cos(y) tan(z)]
```

【实例讲解】 这时的函数 sym 实际是在定义一个符号表达式，这时的符号矩阵中的元素可以是任何的符号或者是表达式，而且长

度没有限制，只是将方括号置于用于创建符号表达式的单引号中。

2.1.4　syms 函数——定义矩阵的又一函数

【语法说明】 syms A1　A2　A3。

【功能介绍】 定义符号矩阵。

【实例 2.4】 syms 和 sym 函数配合使用建立符号矩阵。

```
>> syms  a  b  c ;
>> X1 = sym('Red');
>> X2 = sym(' Blue');
>> X3 = sym('Green')
>> syms_matrix = [a  b  c; X1, X2, X3;int2str([2  6
9])]
syms_matrix =
[  a     b     c]
[Red   Blue  Green]
[  2     6     9]
```

【实例讲解】 先定义矩阵中的每一个元素为一个符号变量，而后像普通矩阵一样输入符号矩阵。

2.1.5　sym 的另一职能——把数值矩阵转化成相应的符号矩阵

【语法说明】 符号矩阵量 = sym（数值矩阵量）。

【功能介绍】 将数值矩阵转化成相应的符号矩阵。

【实例 2.5】 数值矩阵向其对应符号矩阵之间的转化。

```
>> Digit_Matrix = [1/3  sqrt(2)  3.4234;exp(0.23)  log
(29)  23^(-11.23)]
>> Syms_Matrix = sym(Digit_Matrix)
```

上面函数的结果是：

```
Digit_Matrix =
0.3333    1.4142    3.4234
1.2586    3.3673    0.0000
Syms_Matrix =
[          1/3,    sqrt(2),          17117/5000]
```

```
    [5668230535726899*2^(-52),7582476122586655*2^(-51),
5174709270083729*2^(-103)]
```

【实例讲解】 数值型和符号型在 MATLAB 中是不相同的，它们之间不能直接进行转化。所以必须通过 sym 函数来转化。注意：不论矩阵是用分数形式还是浮点形式表示的，将矩阵转化成符号矩阵后，都将以最接近原值的有理数形式或者是函数形式表示。

2.1.6 创建大矩阵

【语法说明】 矩阵变量 =[数值 1 数值 2 … 数值 *n*]。

【功能介绍】 创建大矩阵。

【实例 2.6】 用 M 文件创建大矩阵，文件名为 example.m。

```
exm=[ 123    456   78     9  109   98
45     45    56    2358   186    103
665    7     88    908    51     75
46     8     4589  670    95     56
88     66    99    36     23     2509]
```

在 MATLAB 窗口输入：

```
>>example;
>>size(exm)    %显示 exm 的大小
ans=
5  6           %表示 exm 有 5 行 6 列
```

【实例讲解】 对于大型矩阵，一般创建 M 文件，以便于修改。

2.1.7 cat 函数——创建多维数组

【语法说明】 A=cat(n,A1,A2,…, Am)。

【功能介绍】 创建多维数组。

【实例 2.7】 用 cat 函数建立多维数组。

```
>> A1=[1,2,3;4,5,6;7,8,9];A2=A1';A3=A1-A2;
>> A4=cat(3,A1,A2,A3)
```

上面函数的结果是：

```
A4(:,:,1) =
     1     2     3
```

```
     4     5     6
     7     8     9
A4(:,:,2) =
     1     4     7
     2     5     8
     3     6     9
A4(:,:,3) =
     0    -2    -4
     2     0    -2
     4     2     0
```

【实例讲解】 上例 n=3 构造的是三维数组。当 n=1 和 n=2 时分别构造[A1;A2]和[A1,A2]，都是二维数组。

2.1.8 zeros 函数——零矩阵的生成

【语法说明】

- B = zeros(n)：生成 n×n 全零阵。
- B = zeros(m,n)：生成 m×n 全零阵。
- B = zeros([m n])：生成 m×n 全零阵。
- B = zeros(d1,d2,d3⋯)：生成 d1×d2×d3×⋯全零阵或数组。
- B = zeros([d1 d2 d3⋯])：生成 d1×d2×d3×⋯全零阵或数组。
- B = zeros(size(A))：生成与矩阵 A 相同大小的全零阵。

【功能介绍】 创建零矩阵的几个函数，格式很简单，套用即可。

【实例 2.8】 演示如何生成 n×n、m×n 以及与其余矩阵相同大小的全零矩阵。

```
>> n=6

n =

     6
>> A1 = zeros(n)                    %生成 n×n 全零阵

A1 =

     0     0     0     0     0     0
```

```
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
>> m=4

m =

     4
>> A2 = zeros([m n])          %生成 m×n 全零阵

A2 =

     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0

>> A3 = zeros(size(A1))       %生成与矩阵 A1 相同大小的全零阵

A3 =

     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
```

【实例讲解】 从上面的例子可以看出，用户可以根据需要使用 zeros 函数创建各种维度的零矩阵。

2.1.9 eye 函数——单位矩阵的生成

【语法说明】

- Y=eye(n)：生成 n×n 单位矩阵。
- Y=eye(m,n)：生成 m×n 单位矩阵。
- Y=eye(size(A))：生成与矩阵 A 相同大小的单位矩阵。

【功能介绍】 生成单位矩阵。

【实例 2.9】 演示单位矩阵的生成。

```
>> n=4

n =

     4

>> m=5

m =

     5

>> Y1  =  eye(n)  %生成 n×n 单位矩阵

Y1 =

     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1

>> Y2  =  eye(m,n)  %生成 m×n 单位矩阵

Y2 =

     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
     0     0     0     0

>> Y3  =  eye(size(Y1))  %生成与矩阵 A 相同大小的单位矩阵

Y3 =

     1     0     0     0
     0     1     0     0
```

```
     0     0     1     0
     0     0     0     1
```

【实例讲解】 单位矩阵在 MATLAB 中是十分重要的矩阵类型，在各种分析中都会用到单位矩阵。

2.1.10 ones 函数——生成全 1 阵

【语法说明】

- Y = ones(n)：生成 n×n 全 1 阵。
- Y = ones(m,n)：生成 m×n 全 1 阵。
- Y = ones([m n])：生成 m×n 全 1 阵。
- Y = ones(d1,d2,d3…)：生成 d1×d2×d3×…全 1 阵或数组。
- Y = ones([d1 d2 d3…])：生成 d1×d2×d3×…全 1 阵或数。
- Y = ones(size(A))：生成与矩阵 A 相同大小的全 1 阵。

【功能介绍】 生成全 1 矩阵，以上几种矩阵都比较简单，只要套用函数的语法格式就可以。

【实例 2.10】 演示几种全 1 矩阵的生成。

```
>> n=4

n =

     4

>> m=6

m =

     6

>> X1= ones(m,n)   %生成 m×n 全 1 阵

X1 =

     1     1     1     1
     1     1     1     1
     1     1     1     1
```

```
     1     1     1     1
     1     1     1     1
     1     1     1     1

>> d1=2

d1 =

     2

>> d2=3

d2 =

     3

>> d3=4

d3 =

     4

>> Y = ones(d1,d2,d3)    %生成 d1×d2×d3×…全 1 阵或数组

Y(:,:,1) =

     1     1     1
     1     1     1


Y(:,:,2) =

     1     1     1
     1     1     1


Y(:,:,3) =

     1     1     1
     1     1     1


Y(:,:,4) =
```

```
    1    1    1
    1    1    1
```

【实例讲解】 上例中的Y为三维的矩阵，所以在第3维中的4个模块中分别显示了二维的全2矩阵。

2.1.11 rand函数——生成均匀分布随机矩阵

【语法说明】

- Y = rand(n)：生成 *n* 随机矩阵，其元素在（0,1）内。
- Y = rand(m,n)：生成 m×n 随机矩阵。
- Y = rand([m n])：生成 m×n 随机矩阵。
- Y = rand(m,n,p,···)：生成 m×n×p×···随机矩阵或数组。
- Y = rand([m n p···])：生成 m×n×p×···随机矩阵或数组。
- Y = rand(size(A))：生成与矩阵A相同大小的随机矩阵。
- rand：无变量输入时只产生一个随机数。
- s = rand('state')：产生包括均匀发生器当前状态的35个元素的向量。
- rand('state', s)：使状态重置为s。
- rand('state', 0)：重置发生器到初始状态。
- rand('state', j)：对整数j重置发生器到第j个状态。
- rand('state', sum (100*clock))：每次重置到不同状态。

【功能介绍】 生成各种形式的均匀分布随机矩阵。

【实例2.11】 产生一个3×4随机矩阵。

```
>> R=rand(3,4)
```

上面函数的结果是：

```
R =
    0.9501    0.4860    0.4565    0.4447
    0.2311    0.8913    0.0185    0.6154
    0.6068    0.7621    0.8214    0.7919
```

【实例讲解】 随机矩阵在工程处理和信号处理中有很多的应用。

【实例2.12】 在命令行中产生一个在区间[10, 20]内均匀分布的4阶随机矩阵。

```
>> a=10;b=20;
>> x=a+(b-a)*rand(4)
```

上面函数的结果是：

```
x =
   19.2181   19.3547   10.5789   11.3889
   17.3821   19.1690   13.5287   12.0277
   11.7627   14.1027   18.1317   11.9872
   14.0571   18.9365   10.0986   16.0379
```

【实例讲解】 生成均匀分布的矩阵比较常用，用法相对也很简单，参照上面的语法格式使用即可。

2.1.12 randn 函数——生成正态分布随机矩阵

【语法说明】

- Y = randn(n)：生成 n×n 正态分布随机矩阵。
- Y = randn(m,n)：生成 m×n 正态分布随机矩阵。
- Y = randn([m n])：生成 m×n 正态分布随机矩阵。
- Y = randn(m,n,p,…)：生成 m×n×p×…正态分布随机矩阵或数组。
- Y = randn([m n p…])：生成 m×n×p×…正态分布随机矩阵或数组。
- Y = randn(size(A))：生成与矩阵 A 相同大小的正态分布随机矩阵。
- randn ：无变量输入时只产生一个正态分布随机数。
- s = randn('state')：产生包括正态发生器当前状态的 2 个元素的向量。
- s = randn('state', s)：重置状态为 s。
- s = randn('state', 0)：重置发生器为初始状态。
- s = randn('state', j)：对于整数 j 重置状态到第 j 状态。
- s = randn('state', sum(100*clock))：每次重置到不同状态。

【功能介绍】 产生正态分布随机矩阵。

【实例 2.13】 产生均值为 0.6，方差为 0.1 的 4 阶矩阵。

```
>> mu=0.6; sigma=0.1;
>> x=mu+sqrt(sigma)*randn(4)
```

上面函数得到的结果是：

```
x =
    0.8311    0.7799    0.1335    1.0565
    0.7827    0.5192    0.5260    0.4890
    0.6127    0.4806    0.6375    0.7971
    0.8141    0.5064    0.6996    0.8527
```

【实例讲解】 按照上面例子的格式可以生成任何形式的正态分布矩阵，只需要将均值和方差的数值替换即可。

2.1.13 randperm 函数——产生随机序列

【语法说明】 p = randperm(n)。

【功能介绍】 产生 1～n 之间整数的随机排列。

【实例 2.14】 产生随机数。

```
>> randperm(6)
ans =
     3     2     1     5     4     6
```

【实例讲解】 得到结果中的 6 个整数是系统随机生成的。

2.1.14 linspace 函数——线性等分向量的生成

【语法说明】

- y = linspace(a,b)：在(a, b)上产生 100 个线性等分点。
- y = linspace(a,b,n)：在(a, b)上产生 n 个线性等分点。

【功能介绍】 生成线性等分向量。

【实例 2.15】 在[1 200]之间分别生成 100 个和 20 个等分点，把其等分点的值显示出来。

```
>> a=1

a =
     1

>> b=200
```

```
b =

   200

>> x1 = linspace(a,b) %在(a, b)上产生100个线性等分点

x1 =

  Columns 1 through 8

     1.0000    3.0101    5.0202    7.0303    9.0404
11.0505   13.0606   15.0707

  Columns 9 through 16

    17.0808   19.0909   21.1010   23.1111   25.1212
27.1313   29.1414   31.1515

  Columns 17 through 24

    33.1616   35.1717   37.1818   39.1919   41.2020
43.2121   45.2222   47.2323

  Columns 25 through 32

    49.2424   51.2525   53.2626   55.2727   57.2828
59.2929   61.3030   63.3131

  Columns 33 through 40

    65.3232   67.3333   69.3434   71.3535   73.3636
75.3737   77.3838   79.3939

  Columns 41 through 48

    81.4040   83.4141   85.4242   87.4343   89.4444
91.4545   93.4646   95.4747
  Columns 49 through 56

    97.4848   99.4949  101.5051  103.5152  105.5253
```

```
107.5354  109.5455  111.5556

    Columns 57 through 64

    113.5657  115.5758  117.5859  119.5960  121.6061
123.6162  125.6263  127.6364

    Columns 65 through 72

    129.6465  131.6566  133.6667  135.6768  137.6869
139.6970  141.7071  143.7172

    Columns 73 through 80

    145.7273  147.7374  149.7475  151.7576  153.7677
155.7778  157.7879  159.7980

    Columns 81 through 88

    161.8081  163.8182  165.8283  167.8384  169.8485
171.8586  173.8687  175.8788

    Columns 89 through 96
    177.8889  179.8990  181.9091  183.9192  185.9293
187.9394  189.9495  191.9596

    Columns 97 through 100

    193.9697  195.9798  197.9899  200.0000

  >> x2 = linspace(a,b,20) %在(a, b)上产生20个线性等分点

  x2 =

    Columns 1 through 8

      1.0000   11.4737   21.9474   32.4211   42.8947
53.3684   63.8421   74.3158
    Columns 9 through 16
     84.7895   95.2632  105.7368  116.2105  126.6842
137.1579  147.6316  158.1053
```

```
  Columns 17 through 20

  168.5789  179.0526  189.5263  200.0000
```

【实例讲解】 在[1 200]之间线性分配。

2.1.15　logspace 函数——产生对数等分向量

【语法说明】

- y = logspace(a,b)：在（10^a, 10^b）之间产生 50 个对数等分向量。
- y = logspace(a,b,n)：在（10^a, 10^b）之间产生 *n* 个对数等分向量。

【功能介绍】 产生对数等分向量。

【实例 2.16】 在 $(10^2,10^4)$ 之间分别产生 50 个和 30 个对数的等分向量。

```
    >> a=2
    a =
        2
    >> b=4
    b =
        4
    >> x1 = logspace(a,b)
    x1 =
      1.0e+004 *
      Columns 1 through 8

        0.0100    0.0110    0.0121    0.0133    0.0146
0.0160    0.0176    0.0193

      Columns 9 through 16

        0.0212    0.0233    0.0256    0.0281    0.0309
0.0339    0.0373    0.0409
      Columns 17 through 24
        0.0450    0.0494    0.0543    0.0596    0.0655
0.0720    0.0791    0.0869
```

```
      Columns 25 through 32

         0.0954    0.1048    0.1151    0.1265    0.1389
0.1526    0.1677    0.1842

      Columns 33 through 40

         0.2024    0.2223    0.2442    0.2683    0.2947
0.3237    0.3556    0.3907

      Columns 41 through 48

         0.4292    0.4715    0.5179    0.5690    0.6251
0.6866    0.7543    0.8286

      Columns 49 through 50

         0.9103    1.0000

    >> x2 = logspace(a,b,30)
    x2 =
      1.0e+004 *
      Columns 1 through 8

         0.0100    0.0117    0.0137    0.0161    0.0189
0.0221    0.0259    0.0304

      Columns 9 through 16

         0.0356    0.0418    0.0489    0.0574    0.0672
0.0788    0.0924    0.1083

      Columns 17 through 24

         0.1269    0.1487    0.1743    0.2043    0.2395
0.2807    0.3290    0.3857
      Columns 25 through 30
    0.4520    0.5298    0.6210    0.7279    0.8532
1.0000
```

【实例讲解】 从上面的结果中可以看出，对数等分数据和线性等分数据是不同的。

2.1.16　blkdiag 函数——产生以输入元素为对角线元素的矩阵

【语法说明】 out = blkdiag(a,b,c,d,…)。

【功能介绍】 产生以 a,b,c,d,...为对角线元素的矩阵。

【实例 2.17】 利用 blkdiag 函数生成以 1、2、3、4 为对角线元素的矩阵。

```
>> out = blkdiag(1,2,3,4)
out =
     1     0     0     0
     0     2     0     0
     0     0     3     0
     0     0     0     4
```

【实例讲解】 得到的矩阵以 1、2、3、4 为对角线，其余的元素都为零。

2.1.17　compan 函数——生成友矩阵

【语法说明】 A = compan(u) ：u 为多项式系统向量，A 为友矩阵，A 的第 1 行元素为 -u (2:n)/u(1)，其中 u (2:n)为 u 的第 2 到第 n 个元素，A 为特征值就是多项式的特征根。

【功能介绍】 生成友矩阵。

【实例 2.18】 求多项式 $(x-1)(x+2)(x-3)=x^3-8x+13$ 的友矩阵和根。

```
> u=[1 0 -8 13]
u =
     1     0    -8    13
>> A=compan(u)
A =
     0     8   -13
     1     0     0
     0     1     0
>> eig(A)
ans =
```

```
    -3.4332
     1.7166 + 0.9165i
     1.7166 - 0.9165i
```

【实例讲解】 u 为方程中未知数按降幂排列的系数，eig(A)函数用于求 A 的特征值。

2.1.18 hankel 函数——生成 Hankel 方阵

【语法说明】

- H = hankel(c)：第 1 列元素为 c，反三角以下元素为 0。
- H = hankel(c,r)：第 1 列元素为 c，最后一行元素为 r，如果 c 的最后一个元素与 r 的第一个元素不同，交叉位置元素取 c 的最后一个元素。

【功能介绍】 以上两函数都用来生成 Hankel 方阵。

【实例 2.19】 使用 hankel(c,r)格式建立 Hankel 方阵。

```
c =
     1     2     3     4
r =
     5     6     7     8     9    10
>> h=hankel(c,r)
h =
     1     2     3     4     6     7
     2     3     4     6     7     8
     3     4     6     7     8     9
     4     6     7     8     9    10
```

【实例讲解】 在这个例子可以看出 hankel 矩阵的构成方式，以给定的 c 元素[1 2 3 4]作为第一列和第一行的前半部分，该例中 c 的最后一个元素与 r 的第一个元素不同，所以交叉位置元素取 c 的最后一个元素 4，然后其余的斜对角线取 r 中的元素。

2.1.19 hilb 函数——生成 Hilbert（希尔伯特）矩阵

【背景知识】 Hilbert 矩阵是著名的病态矩阵。MATLAB 中有专门的 Hilbert 矩阵及其准确逆矩阵的生成函数。

【语法说明】 H = hilb(n)：返回 n 阶 Hilbert 矩阵，其元素为

H(i,j)=1/(i+j-1)。

【功能介绍】 产生 Hilbert 矩阵。

【实例 2.20】 产生一个 4 阶 Hilbert 矩阵。

```
>> H=hilb(4)
```

上面函数得到的结果是：

```
H =
    1.0000    0.5000    0.3333    0.2500
    0.5000    0.3333    0.2500    0.2000
    0.3333    0.2500    0.2000    0.1667
    0.2500    0.2000    0.1667    0.1429
```

【实例讲解】 有关病态矩阵的详细信息，请用户查看相应的帮助问题。

2.1.20 invhilb 函数——逆 Hilbert 矩阵生成

【背景知识】 使用一般方法求逆会因为原始数据的微小扰动而产生不可靠的计算结果。MATLAB 中，有一个专门求 Hilbert 矩阵的逆的函数 invhilb(n)。

【语法说明】 H = invhilb(n)：产生 *n* 阶逆 Hilbert 矩阵。

【功能介绍】 求 *n* 阶的 Hilbert 矩阵的逆矩阵。

【实例 2.21】 产生一个 4 阶逆 Hilbert 矩阵。

```
>> H = invhilb(4)
```

上面函数得到的结果为：

```
H =
          16        -120         240        -140
        -120        1200       -2700        1680
         240       -2700        6480       -4200
        -140        1680       -4200        2800
```

【实例讲解】 用户可以直接尝试使用 inv 函数求解 Hilbert 矩阵的逆矩阵。

2.1.21 pascal 函数——生成 Pascal 矩阵

【背景知识】 通常，二次项展开后的系数随 *n* 的增大组成一

个三角形表，称为杨辉三角形。由杨辉三角形表组成的矩阵称为帕斯卡（Pascal）矩阵，函数 pascal(n)生成一个 *n* 阶帕斯卡矩阵。

【语法说明】

- A = pascal(n)：产生 *n* 阶 Pascal 矩阵，它是对称、正定矩阵，它的元素由 Pascal 三角组成，它的逆矩阵的所有元素都是整数。
- A = pascal(n,1)：返回由下三角的 Cholesky 系数组成的 Pascal 矩阵。
- A = pascal(n,2)：返回 Pascal(n,1)的转置和交换的形式。

【功能介绍】 生成 Pascal 矩阵。

【实例 2.22】 分别产生 4 阶 Pascal 矩阵、由下三角的 Cholesky 系数组成的 Pascal 矩阵以及该矩阵的转置形式。

```
>> A=pascal(4)
A =
     1     1     1     1
     1     2     3     4
     1     3     6    10
     1     4    10    20
>> A=pascal(3,1)
A =
     1     0     0
     1    -1     0
     1    -2     1
>> A=pascal(3,2)
A =
     1     1     1
    -2    -1     0
     1     0     0
```

【实例讲解】 这个例子中，第一个函数产生了一个 4 阶 Pascal 矩阵，它是对称、正定矩阵，矩阵中的元素满足杨辉三角形法则；第二个函数产生了一个下三角的 Cholesky 系数组成的 Pascal 矩阵；可以看出，第三个函数得到的结果刚好是第二个矩阵的转置矩阵。

2.1.22 toeplitz 函数——生成托普利兹矩阵

【背景知识】 托普利兹（Toeplitz）矩阵中，除第一行第一列

外，其他每个元素都与左上角的元素相同。

【语法说明】

- T = toeplitz(c,r)：生成一个非对称的托普利兹矩阵，将 c 作为第 1 列，将 r 作为第 1 行，其余元素与左上角相邻元素相等。
- T = toeplitz(r)：用向量 r 生成一个对称的托普利兹矩阵。

【功能介绍】 产生托普利兹矩阵。

【实例 2.23】 对于给定的向量 *c*、*r* 建立一个非对称的托普利兹矩阵。

```
>>c=[1 2 3 4 5];
>>  r=[1.3 2.4 3.5 4.6 5.7];
>>  T=toeplitz(c,r)
T =
    1.0000    2.4000    3.5000    4.6000    5.7000
    2.0000    1.0000    2.4000    3.5000    4.6000
    3.0000    2.0000    1.0000    2.4000    3.5000
    4.0000    3.0000    2.0000    1.0000    2.4000
    5.0000    4.0000    3.0000    2.0000    1.0000
```

【实例讲解】 上例中将向量 *c* 作为托普利兹矩阵的第一列，将向量 *r* 作为第一行，其余元素按照左上角相邻元素确定。

2.1.23 wilkinson 函数——生成 Wilkinson 特征值测试阵

【语法说明】 W=wilkinson(n)返回 *n* 阶 Wilkinson 特征值测试阵。

【功能介绍】 产生 Wilkinson 特征值测试阵。

【实例 2.24】 分别建立 4 阶和 7 阶 Wilkinson 特征值测试阵。

```
>> W=wilkinson(4)
W =
     3/2          1            0            0
      1          1/2           1            0
      0           1           1/2           1
      0           0            1           3/2
>> W=wilkinson(7)
W =
      3     1     0     0      0      0      0
      1     2     1     0      0      0      0
```

```
     0     1     1     1     0     0     0
     0     0     1     0     1     0     0
     0     0     0     1     1     1     0
     0     0     0     0     1     2     1
     0     0     0     0     0     1     3
```

【实例讲解】 关于特征值测试矩阵的具体内容，请用户查看相应的资料。

2.2 矩阵的运算

矩阵的运算是数值运算中极为关键和重要的部分，MATLAB 的强大功能为矩阵的运算提供了极丰富和完善的函数库，在下面的小节中将进行详细的介绍和说明。

2.2.1 矩阵的加减运算指令

【语法说明】 “+”和“-”分别为矩阵的加、减运算指令。

【功能介绍】 其功能是将矩阵对应元素相加、减，即按线性代数中矩阵的“+”，“-”运算进行。

【实例 2.25】 计算两个矩阵相加和相减结果。

```
>>A=[1, 1, 1; 1, 2, 3; 1, 3, 6]
>>B=[8, 1, 6; 3, 5, 7; 4, 9, 2]
>>sum1=A+B
>>sum2=A-B
```

计算结果为：

```
sum1=
9   2   7
4   7  10
5  12   8
sum2=
-7     0   -5
-2    -3   -4
-3    -6    4
```

【实例讲解】 上例中得到的结果分别是矩阵 ***A*** 和矩阵 ***B*** 中的对应元素相加减的结果。

2.2.2 矩阵的简单乘法

【语法说明】 “*”为矩阵的乘法运算指令。

【功能介绍】 其功能是将两个矩阵作数值运算中的乘法处理。

【实例 2.26】 计算两个简单矩阵的乘积。

```
>>X= [2  3  4  5;
      1  2  2  1];
>>Y=[0  1  1;
     1  1  0;
     0  0  1;
     1  0  0];
Z=X*Y
```

上面函数的结果显示为：

```
Z=
   8  5  6
   3  3  3
```

【实例讲解】 在 MATLAB 中两个矩阵相乘的计算方法与数值计算中完全相同，即放在前面的矩阵的各行元素，分别与放在后面的矩阵的各列元素对应相乘并相加，得到的结果即为新的乘积矩阵。

【语法说明】 “*”也作为矩阵与数的乘法运算指令。

【功能介绍】 其功能是将一个矩阵与数值作乘法运算。

【实例 2.27】 计算矩阵与数值的乘积。

```
>> A=[1,3,5;2,4,7;9,8,6]
A =
     1     3     5
     2     4     7
     9     8     6
>> A*2
```

结果显示为：

```
ans =
     2     6    10
```

```
    4    8    14
   18   16    12
```

【实例讲解】 在 MATLAB 中矩阵与数相乘的计算方法与数值计算中完全相同，即矩阵中的每一个元素分别与该数相乘作为新矩阵的元素，得到的结果即为新的乘积矩阵。

2.2.3 dot 函数——向量的点积

【语法说明】

- C = dot(A,B)：若 A、B 为向量，则返回向量 A 与 B 的点积，A 与 B 长度相同；若为矩阵，则 A 与 B 有相同的维数。
- C = dot(A,B,dim)：在 dim 维数中给出 A 与 B 的点积。

【功能介绍】 其功能是向量的点乘（内积），即维数相同的两个向量的点乘。

【实例 2.28】 计算向量的内积。

```
>> X=[1,3,2];
>> Y=[2,6,3];
>> Z=dot(X,Y)
Z =
    26
```

【实例讲解】 在 MATLAB 中向量的点乘与线性代数计算中完全相同，即两个向量中的元素对应相乘并相加求和。还可以用.*代替 dot。

2.2.4 cross 函数——向量叉乘

【语法说明】

- C = cross(A,B)：若 A、B 为向量，则返回 A 与 B 的叉乘，即 C=A×B，A、B 必须是 3 个元素的向量；若 A、B 为矩阵，则返回一个 3×n 矩阵，其中的列是 A 与 B 对应列的叉积，A、B 都是 3×n 矩阵。
- C = cross(A,B,dim)：在 dim 维数中给出向量 A 与 B 的叉积。A 和 B 必须具有相同的维数，size(A,dim)和 size(B,dim)必须是 3。

【功能介绍】 用于求两个向量的叉乘运算。

【实例 2.29】 计算垂直于向量(1, 2, 3)和(4, 5, 6)的向量。

```
>>a=[1  2  3];
>>b=[4  5  6];
>>c=cross(a,b)
```

计算结果如下：

```
c=
    -3     6    -3
```

【实例讲解】 可得垂直于向量(1, 2, 3)和(4, 5, 6)的向量为±(-3, 6, -3)。在数学上，两向量的叉乘是一个过两相交向量的交点且垂直于两向量所在平面的向量。在 MATLAB 中，用函数 cross 实现。

2.2.5 向量的混合积运算

【语法说明】 混合积由 dot 函数和 cross 函数实现。

【功能介绍】 实现向量的混合积运算。

【实例 2.30】 计算向量 $a=(1, 2, 3)$、$b=(4, 5, 6)$和 $c=(-1, -2, -3)$的混合积 $a\cdot(b\times c)$。

```
>> a=[1  2  3];
>>b=[4  5  6];
>>c=[-7  -2  -3];
>> d=dot(a, cross(b, c))
```

上面函数的结果为：

```
d =
    18
```

【实例讲解】 先叉乘后点乘，顺序不可颠倒。

2.2.6 conv 函数——矩阵的卷积和多项式乘法

【语法说明】 w = conv(u,v) ：u、v 为向量，其长度可不相同。

【功能介绍】 实现向量卷积运算。

【实例 2.31】 展开多项式$(s^2+2s+2)(s+4)(s+1)$。

```
>> w=conv([1,2,2],conv([1,4],[1,1]))
w =
     1     7    16    18     8
>> P=poly2str(w,'s')     %将 w 表示成多项式
```

```
P =
  s^4 + 7 s^3 + 16 s^2 + 18 s + 8
```

【实例讲解】 长度为 m 的向量序列 u 和长度为 n 的向量序列 v 的卷积（Convolution）定义为：$w(k)=\sum_{j=1}^{k}u(j)v(k+1-j)$。式中：$w$ 向量序列的长度为$(m+n-1)$，当 $m=n$ 时，

```
w(1) = u(1)*v(1)
w(2) = u(1)*v(2)+u(2)*v(1)
w(3) = u(1)*v(3)+u(2)*v(2)+u(3)*v(1)
...
w(n) = u(1)*v(n)+u(2)*v(n-1)+ … +u(n)*v(1)
...
w(2*n-1) = u(n)*v(n)
```

2.2.7 deconv 函数——反褶积（解卷）和多项式除法运算

【语法说明】 [q,r] = deconv(v,u)：多项式 v 除以多项式 u，返回商多项式 q 和余多项式 r。

【功能介绍】 实现向量反褶积（解卷）运算。

【实例 2.32】 求多项式 $(x^3+2x^2+3x+4)(10x^2+20x+30)$ 的反褶积。

```
>>u = [1   2   3   4]
>>v = [10   20   30]
>>c = conv(u,v)
c =
   10    40    100    160    170    120
```

则反褶积为：

```
>>[q,r] = deconv(c,u)
```

上面函数的结果为：

```
q =
   10    20    30
r =
   0    0    0    0    0    0
```

【实例讲解】 v、u、q、r 都是按降幂排列的多项式系数向量。

2.2.8 kron 函数——张量积

【语法说明】 C=kron (A,B) ：A 为 $m \times n$ 矩阵，B 为 $p \times q$ 矩阵，则 C 为 $mp \times nq$ 矩阵。

【功能介绍】 实现向量张量积运算。

【实例 2.33】 $\boldsymbol{A}=\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$， $\boldsymbol{B}=\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{bmatrix}$， 求 $\boldsymbol{A}\otimes\boldsymbol{B}$。

```
>> A=[1 2;3 4];B=[1 2 3;4 5 6;7 8 9];
>> C=kron(A,B)
```

上面函数的结果为：

```
C =
     1     2     3     2     4     6
     4     5     6     8    10    12
     7     8     9    14    16    18
     3     6     9     4     8    12
    12    15    18    16    20    24
    21    24    27    28    32    36
```

【实例讲解】 $\boldsymbol{A}$ 与 $\boldsymbol{B}$ 的张量积定义为：$\boldsymbol{C}=\boldsymbol{A}\otimes\boldsymbol{B}=\begin{bmatrix}a_{11}B & a_{12}B & \cdots & a_{1n}B\\ a_{21}B & a_{22}B & \cdots & a_{2n}B\\ \vdots & \vdots & \ddots & \vdots\\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B\end{bmatrix}$，$\boldsymbol{A}\otimes\boldsymbol{B}$ 与 $\boldsymbol{B}\otimes\boldsymbol{A}$ 均为 $mp \times nq$ 矩阵，但一般地 $\boldsymbol{A}\otimes\boldsymbol{B}\neq\boldsymbol{B}\otimes\boldsymbol{A}$。所以在上例中求出的结果并不等于 $\boldsymbol{B}\otimes\boldsymbol{A}$。

2.2.9 intersect 函数——求两个集合的交集

【语法说明】

- c = intersect(a,b) ：返回向量 a、b 的公共部分。
- c = intersect(A,B,'rows') ：A、B 为相同列数的矩阵，返回元素相同的行。
- [c,ia,ib] = intersect(a,b) ：c 为 a、b 的公共元素，ia 表示公共元素在 a 中的位置，ib 表示公共元素在 b 中位置。

【功能介绍】 实现两个集合的交集。

【实例 2.34】 求两个向量 *A*、*B* 的交集。

```
>> A=[1 2 3 4;1 2 4 6;6 7 1 4]
A =
     1     2     3     4
     1     2     4     6
     6     7     1     4
>> B=[1 2 3 8;1 1 4 6;6 7 1 4]
B =
     1     2     3     8
     1     1     4     6
     6     7     1     4
>> C=intersect(A,B,'rows')
```

得到 *A*、*B* 的交集为：

```
C =
     6     7     1     4
```

【实例讲解】 这个例子具有相同的列数，返回的为相同的行元素。

【实例 2.35】 求两个向量 *A*、*B* 的交集，并找出相同元素分别在两个矩阵中的位置。

```
>> A = [1 2 3 4]; B = [1 2 3 4 5 6 7 8];
>> [c,ia,ib] = intersect(A,B)
```

得到的结果为：

```
c =
     1     2     3     4
ia =
     1     2     3     4
ib =
     1     2     3     4
```

【实例讲解】 返回相同元素，并且返回在两个矩阵中的位置。

2.2.10 ismember 函数——检测集合中的元素

【语法说明】

- k = ismember(a,S)：当 a 中元素属于 S 时，k 取 1，反之，k 取 0。

- k = ismember(A,S,'rows')：A、S 有相同的列，返回行相同 k 取 1，不相同取 0 的列向量。

【功能介绍】 实现两个集合的交集。

【实例 2.36】 判断向量 *a* 中的元素是否在向量 *S* 中。

```
>> S=[0 2 4 6 8 10 12 14 16 18 20];
>> a=[1 2 3 4 5 6];
>> k=ismember(a,S)
```

上面函数的结果为：

```
k =
     0     1     0     1     0     1         %1表示相同元素的位置
```

【实例讲解】 $k = 0$ 的位置表示 *a* 中的该元素不在 *S* 中；$k = 1$ 的位置表示 *a* 中的该元素在 *S* 中。

2.2.11 setdiff 函数——求两集合的差

【语法说明】

- c = setdiff(A,B)：返回属于 A 但不属于 B 的不同元素的集合，C = a-b。
- c = setdiff(A,B,'rows')：返回属于 A 但不属于 B 的不同行。
- [c,i] = setdiff(…)：c 与前面一致，i 表示 c 中元素在 A 中的位置。

【功能介绍】 求两个集合的差。

【实例 2.37】 求属于 A 但不属于 B 的不同元素的集合。

```
>> A = [1 7 9 6 20]; B = [1 2 3 4 6 104 2];
>> C=setdiff(A,B)
C =
     7     9
```

【实例讲解】 得到的结果是具体的每一个元素。

【实例 2.38】 求属于 A 但不属于 B 的不同行。

```
>> A=[1 2 3 4;1 2 4 6;6 7 1 4]
>> B=[1 2 3 8;1 1 4 6;6 7 1 4]
>> c=setdiff(A,B,'rows')
```

上面函数的结果为：

```
c =
     1     2     3     4
     1     2     4     6
```

【实例讲解】 得到的结果是行元素，而不是具体到每一个元素值。

2.2.12 setxor 函数——求两个集合交集的非（异或）

【语法说明】

- c = setxor(a,b) ：返回集合 a、b 交集的非。
- c = setxor(A,B,'rows')：返回矩阵 A、B 交集的非，A、B 有相同列数。
- [c,ia,ib] = setxor(…) ：ia、ib 表示 c 中元素分别在 a（或 A）、b（或 B）中位置。

【功能介绍】 求两个集合的交集的非。

【实例 2.39】 已知向量 A=[1,2,3,4]和向量 B=[2,4,5,8]，求 A、B 交集的非。

```
>> A=[1  2  3  4];
>> B=[2  4  5  8];
>> C=setxor(A,B)
```

上面函数的结果为：

```
C =
     1     3     5     8
```

【实例讲解】 得到的结果是具体的每一个元素。

【实例 2.40】 已知向量$A=\begin{bmatrix}1 & 2 & 3 & 4\\1 & 2 & 4 & 6\\6 & 7 & 1 & 4\end{bmatrix}$和向量$B=\begin{bmatrix}1 & 2 & 3 & 8\\1 & 1 & 4 & 6\\6 & 7 & 1 & 4\end{bmatrix}$，求 A，B 交集的非。

```
>> A=[1 2 3 4;1 2 4 6;6 7 1 4]
A =
     1     2     3     4
     1     2     4     6
     6     7     1     4
```

```
>> B=[1 2 3 8;1 1 4 6;6 7 1 4]
B =
     1     2     3     8
     1     1     4     6
     6     7     1     4
>> [C,ia,ib]=setxor(A,B,'rows')
```

上面函数得到的结果为：

```
C =
     1     1     4     6
     1     2     3     4
     1     2     3     8
     1     2     4     6
ia =
     1
     2
ib =
     2
     1
```

【实例讲解】 得到的结果是具体行元素和行元素所在的位置。

2.2.13 union 函数——求两集合的并集

【语法说明】

- c = union(a,b)：返回 a、b 的并集，即 c = a∪b。
- c = union(A,B,'rows')：返回矩阵 A、B 不同行向量构成的大矩阵，其中相同行向量只取其一。
- [c,ia,ib] = union(…)：ia、ib 分别表示 c 中行向量在原矩阵（向量）中的位置。

【功能介绍】 求两个集合的并集。

【实例 2.41】 已知向量 *A*=[1,2,3,4]和向量 *B*=[5,6,7,8,4,3]，求 *A*，*B* 集合的并集。

```
>> A=[1 2 3 4];
>> B=[5 6 7 8 4 3];
>> c=union(A,B)
```

则结果为：

```
c =
     1     2     3     4     5     6     7     8
```

【实例讲解】 得到的结果是 ***A***、***B*** 两个矩阵中所有元素的集合，如果有元素是重复的，那么只取一次。

【实例 2.42】 已知向量$A=\begin{bmatrix}1 & 2 & 3 & 4\\2 & 3 & 4 & 5\end{bmatrix}$和向量$B=\begin{bmatrix}1 & 2 & 3 & 8\\1 & 1 & 4 & 6\end{bmatrix}$，求 *A*，*B* 并集。

```
>>A=[1 2 3 4;1 2 4 6]
A=
     1     2     3     4
     2     3     4     5
>> B=[1 2 3 8;1 1 4 6]
B =
     1     2     3     8
     1     1     4     6
>> [c,ia,ib]=union(A,B,'rows')
c =
     1     1     4     6
     1     2     3     4
     1     2     3     8
     2     3     4     5
ia =
     1
     2
ib =
     2
     1
```

【实例讲解】 得到的结果是 ***A***、***B*** 两个矩阵的行合并的结果，组合的顺序是按照每一行的前面元素的顺序，如果前面的几个元素相同，再依次往后比较。ia、ib 分别表示了在 ***C*** 矩阵中原矩阵的行元素的位置。

2.2.14 unique 函数——取集合的单值元素

【语法说明】

- b = unique (a)：取集合 a 的不重复元素构成的向量。

- b = unique (A,'rows')：返回 A、B 不同行元素组成的矩阵。
- [b,i,j] = unique (…)：i、j 体现 b 中元素在原向量（矩阵）中的位置。

【功能介绍】 求两个集合的并集。

【实例 2.43】 已知向量 A=[1,1,2,2,4,4,6,4,4]，求 A 集合中的单值元素。

```
>> A=[1 1 2 2 4 4 6 4 6]
A =
     1     1     2     2     4     4     6     4     6
>> [c,i,j]=unique(A)
c =
     1     2     4     6
i =
     2     4     8     9
j =
  1     1     2     2     3     3     4     3     4
```

【实例讲解】 c 表示单值元素有哪些，依次列出来，i 表示这些单值元素最后一次出现在矩阵中的位置。

【实例 2.44】 已知向量 $A=\begin{bmatrix}1 & 2 & 2 & 4\\1 & 1 & 4 & 6\\1 & 1 & 4 & 6\end{bmatrix}$，求 A 集合中的单值元素及行列位置。

```
>> A=[1 2 2 4;1 1 4 6;1 1 4 6]
A =
  1     2     2     4
     1     1     4     6
     1     1     4     6
>> [c,i,j]=unique(A,'rows')
c =
     1     1     4     6
     1     2     2     4
i =
     3
     1
j =
```

```
2
1
1
```

【实例讲解】 i、j 体现了 c 中元素在原向量（矩阵）中的位置。

2.2.15 矩阵的除法运算

【语法说明】 Matlab 提供了两种除法运算：左除（\）和右除（/）。一般情况下，x=a\b 是方程 a*x =b 的解，而 x=b/a 是方程 x*a=b 的解。

【功能介绍】 求两个矩阵的除法。

【实例 2.45】 求矩阵 a 与矩阵 b 的除法运算。

```
a=[1  2  3; 4  2  6; 7  4  9]
b=[4; 1; 2];
x=a\b
```

结果为：

```
x=
-1.5000
2.0000
0.5000
```

【实例讲解】 如果 a 为非奇异矩阵，则 a\b 和 b/a 可通过 a 的逆矩阵与 b 阵得到：

```
a\b = inv(a)*b
b/a = b*inv(a)
```

在数组除法运算中，A./B 表示 A 中元素与 B 中元素对应相除。

2.2.16 矩阵乘方

【语法说明】 “^”为矩阵乘方运算符。

- 当 A 为方阵，P 为大于 0 的整数时，A^P 表示 A 的 P 次方，即 A 自乘 P 次；P 为小于 0 的整数时，A^P 表示 A^{-1} 的 P 次方。
- 当 A 为方阵，p 为非整数时，则 $A \wedge P = V\begin{bmatrix} d_{11}^{p} & & \\ & \ddots & \\ & & d_{nn}^{p} \end{bmatrix}V^{-1}$，

其中 V 为 A 的特征向量，$\begin{bmatrix} d_{11} & & \\ & \ddots & \\ & & d_{nn} \end{bmatrix}$为特征值对角矩阵。如果有重根，以上指令不成立。

- 标量的矩阵乘方 P^A，标量的矩阵乘方定义为 $P^A = V\begin{bmatrix} p^{d_{11}} & & \\ & \ddots & \\ & & p^{d_{nn}} \end{bmatrix}V^{-1}$，式中 V，D 取自特征值分解 AV=AD。
- 标量的数组乘方 $P.\hat{}A$，标量的数组乘方定义为 $P.\hat{}A = \begin{bmatrix} p^{a_{11}} & \cdots & p^{a_{1n}} \\ \vdots & & \vdots \\ p^{a_{m1}} & \cdots & p^{a_{mn}} \end{bmatrix}$，数组乘方 $A.\hat{}P$ 表示 A 的每个元素的 P 次乘方。

【功能介绍】 求矩阵的乘方。

【实例 2.46】 求矩阵 *A* 的乘方运算，包括上面介绍的各种形式。

```
>> A=[1,2,3;4,5,6;9,6,12]
A =
     1     2     3
     4     5     6
     9     6    12
>> A^5
ans =
      190224      162882      277857
      426366      365085      622782
      766503      656316     1119609
>> A^-2
ans =
    0.8272   -0.1235   -0.1358
   -0.0988    0.3580   -0.1728
   -0.5062   -0.1235    0.1975
>> A^0.5
ans =
   0.4420 + 0.9342i   0.4992 - 0.1764i   0.6529 - 0.1337i
```

```
   0.7109 + 0.0398i   1.8716 - 0.0075i   1.1142 - 0.0057i
   2.1504 - 0.6629i   1.0184 + 0.1252i   3.0916 + 0.0949i
>> 3.^A
ans =
           3           9          27
          81         243         729
       19683         729      531441
>> A.^3
ans =
           1           8          27
          64         125         216
         729         216        1728
```

【实例讲解】 注意 3.^A 与 A.^3 的区别。

2.2.17 expm 函数——方阵指数函数

【语法说明】

- Y = expm(A)：使用 Pade 近似算法计算 e^A，这是一个内部函数，A 为方阵。
- Y=expm1(A)：使用一个 M 文件和内部函数相同的算法计算 e^A。
- Y=expm2(A)：使用泰勒级数计算 e^A。
- Y=expm3(A)：使用特征值和特征向量计算 e^A。

【功能介绍】 求方阵的指数。

【实例 2.47】 求方阵 ***A*** 的指数函数。

```
>> A=[1,2,3;4,5,6;9,6,12]
A=
     1     2     3
     4     5     6
     9     6    12
>> Y=expm(A)
Y =
   1.0e+007 *
    0.4856    0.4158    0.7093
    1.0885    0.9320    1.5899
    1.9568    1.6755    2.8582
>> Y=expm1(A)
```

```
Y =
  1.0e+005 *
    0.0000    0.0001    0.0002
    0.0005    0.0015    0.0040
    0.0810    0.0040    1.6275
```

【实例讲解】 对 e 分别求 1，2，3，4，5，6，7，8，9 次方，并按照 3×3 的矩阵组成新的矩阵即为结果。上例中的两个函数求解方法不同。

2.2.18 logm 函数——求矩阵的对数

【语法说明】

- Y = logm(X)：计算矩阵 X 的对数，它是 expm(X)的反函数。
- [Y,esterr] = logm(X)：esterr 为相对残差的估计值，norm(expm(Y)-X)/norm(X)。

【功能介绍】 求矩阵的对数。

【实例 2.48】 先对方阵 ***A*** 运用指数运算，然后利用 logm 求这个矩阵的对数。

```
>> A=[1 1 0;0 0 2;0 0 -1];
>> Y=expm(A)
Y =
   2.7183    1.7183    1.0862
        0    1.0000    1.2642
        0         0    0.3679
>> A=logm(Y)
A =
   1.0000    1.0000    0.0000
        0         0    2.0000
        0         0   -1.0000
```

【实例讲解】 对矩阵的每个元素先求幂次方，再对新的矩阵求对数，经过两次可逆运算后，得到的结果与原矩阵相同。

2.2.19 funm 函数——方阵的函数运算

【语法说明】

- F = funm(A,fun) ：A 为方阵，计算由 fun 指定的 A 的矩阵

函数，fun 可以是任意基本函数，如 sin、cos 等。

- [F,esterr] = funm(A,fun) ：esterr 为结果所产生的相对误差的估计值。

【功能介绍】 求方阵的任何基本数学函数。

【实例 2.49】 求方阵$\begin{bmatrix}1 & 3 & 5\\ 2 & 5 & 8\\ 4 & 6 & 9\end{bmatrix}$的各种基本数学函数。

```
>> A=[1,3,5;2,5,8;4,6,9]
A =
     1     3     5
     2     5     8
     4     6     9
>> F=funm(A,'sin')
F =
   -0.5701   -0.0548    0.2165
   -0.6523   -0.0275    0.1617
    0.5427   -0.1096   -0.4084
>> F2=funm(A,'cos')
F2 =
    0.4885   -0.3726   -0.4245
   -0.6416    0.3847   -0.7971
   -0.1037   -0.7453   -0.3087
>> F3=funm(A,'exp')
F3 =
  1.0e+006 *
    0.8652    1.5772    2.4424
    1.4239    2.5957    4.0196
    1.7304    3.1544    4.8848
```

【实例讲解】 上例中只以 3 个比较基本的函数作为例子，所有的基本函数都可以以这种方式作用于矩阵，相当于矩阵中的每一个元素都作了基本函数操作。

2.2.20 sqrtm 函数——矩阵的方根

【语法说明】

- X = sqrtm(A)：矩阵 A 的平方根 $A^{1/2}$，相当于 X*X=A，求

X。若 A 的特征值有非负实部，则 X 是唯一的；若 A 的特征值有负的实部，则 X 为复矩阵；若 A 为奇异矩阵，则 X 不存在。

- [X,resnorm] = sqrtm(A) ：resnorm 为结果产生的相对误差。
- [X,alpha,condest] = sqrtm(A)： alpha 为稳定因子，condest 为结果的条件数的估计值。

【功能介绍】 求矩阵的方根。

【实例 2.50】 对矩阵 $\boldsymbol{A}=\begin{bmatrix}1 & 4\\3 & 9\end{bmatrix}$求方根。

```
>> A =[1,4;3,9]
A =
     1     4
     3     9
>> B= sqrtm(A)
B =
   0.3915 + 0.4740i   1.2125 - 0.2041i
   0.9094 - 0.1531i   2.8165 + 0.0659i
```

【实例讲解】 $\boldsymbol{B}$ 为复数矩阵，$\boldsymbol{B}\times\boldsymbol{B}=\boldsymbol{A}$。

2.2.21 polyvalm 函数——求矩阵的多项式

【语法说明】 polyvalm(P, A)：%P 为多项式系数向量，方阵 A 为多项式变量，返回多项式值。

【功能介绍】 求矩阵的多项式。

2.2.22 矩阵转置

【语法说明】 “′ ”运算符为矩阵转置运算符。

- 若矩阵 $\boldsymbol{A}$ 的元素为实数，则与线性代数中矩阵的转置相同。
- 若 $\boldsymbol{A}$ 为复数矩阵，则 $\boldsymbol{A}$ 转置后的元素由 $\boldsymbol{A}$ 对应元素的共轭复数构成。
- 若仅希望转置，则用如下函数：A.′ 。

【功能介绍】 求矩阵的转置。

【实例2.51】 求矩阵$A=\begin{bmatrix}1 & 3 & 5\\4 & 5 & 6\\7 & 8 & 9\end{bmatrix}$和$B=\begin{bmatrix}1+2i & 2+3i\\i & 1-3.4i\end{bmatrix}$的转置。

```
A =
     1     3     5
     4     5     6
     7     8     9
>> A'
ans =
     1     4     7
     3     5     8
     5     6     9
>> B=[1+2*i,2+3*i;i,1-3.4*i]
B =
   1.0000 + 2.0000i   2.0000 + 3.0000i
        0 + 1.0000i   1.0000 - 3.4000i
>> B'
ans =
   1.0000 - 2.0000i        0 - 1.0000i
   2.0000 - 3.0000i   1.0000 + 3.4000i
```

【实例讲解】 对于实数矩阵转置就是行的元素变成对应列的元素，而对于复数矩阵则是行列元素互换的同时，复数换作其对应的共轭矩阵。

2.2.23 det 函数——求方阵的行列式

【语法说明】 d = det(X)：返回方阵 X 的多项式的值。

【功能介绍】 求方阵的行列式。

【实例 2.52】 输入数值矩阵，然后换解行列式。

```
>> A=[1 2 3;4 5 6;7 8 9]
A =
   1     2     3
   4     5     6
   7     8     9
>> D=det(A)
D =
     0
```

【实例讲解】 与数值计算中的一样，用来算方阵的行列式。

2.2.24　inv 函数——求矩阵的逆

【语法说明】 Y=inv(X)：求方阵 X 的逆矩阵。若 X 为奇异阵或近似奇异阵，将给出警告信息。

【功能介绍】 求方阵的逆矩阵。

【实例 2.53】 求 $\boldsymbol{A}=\begin{pmatrix}1 & 2 & 3\\ 4 & 8 & 5\\ 2 & 9 & 12\end{pmatrix}$ 的逆矩阵。

函数直接求逆：

```
>> A=[1  2  3; 4  8  5; 2  9  12];
>> Y=inv(A)
```

则结果显示为：

```
 Y =
    1.4571    0.0857   -0.4000
   -1.0857    0.1714    0.2000
    0.5714   -0.1429    0
```

方法二：

由增广矩阵 $\boldsymbol{B}=\begin{pmatrix}1 & 2 & 3 & 1 & 0 & 0\\ 4 & 8 & 5 & 0 & 1 & 0\\ 2 & 9 & 12 & 0 & 0 & 1\end{pmatrix}$ 进行初等行变换：

```
>>B=[1, 2, 3, 1, 0, 0; 2, 2, 1, 0, 1, 0; 3, 4, 3, 0, 0, 1];
>>C=rref(B)      %将行化成最简形式
>>X=C(:, 4:6)     %取矩阵 C 中的 A^(-1)部分
```

【实例讲解】 上面讲述的方法与线性代数中的原理相同。

2.2.25　pinv 函数——求矩阵的伪逆矩阵

【语法说明】

- B = pinv(A)：求矩阵 A 的伪逆矩阵。
- B = pinv(A, tol)：tol 为误差。

【功能介绍】 求方阵的伪逆矩阵。

【实例 2.54】 对 5 阶魔方阵进行求伪逆。

```
>> A=magic(5);    %产生 5 阶魔方阵
>> A=A(:,1:4)     %取 5 阶魔方阵的前 4 列元素构成矩阵 A
A =
    17    24     1     8
    23     5     7    14
     4     6    13    20
    10    12    19    21
    11    18    25     2
>> X=pinv(A)     %计算 A 的伪逆
X =
   -0.0041    0.0527   -0.0222   -0.0132    0.0069
    0.0437   -0.0363    0.0040    0.0033    0.0038
   -0.0305    0.0027   -0.0004    0.0068    0.0355
    0.0060   -0.0041    0.0314    0.0211   -0.0315
```

【实例讲解】 当矩阵为长方阵时，方程 AX=I 和 XA=I 至少有一个无解，这时 A 的伪逆能在某种程度上代表矩阵的逆，若 A 为非奇异矩阵，则 pinv(A) = inv(A)。

2.2.26 trace 函数——矩阵的迹

【语法说明】 b=trace (A)：返回矩阵 A 的迹，即 A 的对角线元素之和。

【功能介绍】 求矩阵的迹。

【实例 2.55】 求矩阵 $A=\begin{bmatrix}3 & 5 & 7\\4 & 6 & 12\\1 & 9 & 5\end{bmatrix}$ 的迹。

```
>> A=[3 5 7;4 6 12;1 9 5]
A =
     3     5     7
     4     6    12
     1     9     5
>> B=trace(A)
B =
    14
```

【实例讲解】 从上面的结果中可以看出，trace 函数求解的结果符合矩阵迹的定义。

2.2.27 norm 函数——求矩阵和向量的范数

【语法说明】

- n = norm(X)：求向量 X 的欧几里德范数，即 $\|X\|_2=\sqrt{\sum|x_k|^2}$。
- n = norm(X,inf)：求向量 X 的 ∞-范数，即 $\|X\|=\max(abs(X))$。
- n = norm(X,1)：求向量 X 的 1-范数，即 $\|X\|_1=\sum_k|x_k|$。
- n = norm(X,-inf)：求向量 X 的元素的绝对值的最小值，即 $\|X\|=\min(abs(X))$。
- n = norm(X, p)：求向量 X 的 p-范数，即 $\|X\|_p=\sqrt[p]{\sum_k|x_k|^p}$，

所以 norm(X,2) = norm(X)。

【功能介绍】 求向量的范数。

【实例 2.56】 求向量 $X=[1,2,3,6,9]$的各种范数。

```
>> X =[1,2,3,6,9]
X =
    1     2     3     6     9
>>n1 =  norm(X)
 n1=
   11.4455
>> n2 =norm(X,inf)
 n2=
    9
>> n3 = norm(X,1)
n3 =
   21.0000
>> n4 = norm(X,-inf)
n4 =
    1
```

```
>> n5 = norm(X,3)
n5 =
   9.9363
```

【实例讲解】 分别按照【语法说明】中的讲解方法求取向量 A 的各种范数。

【语法说明】

- n = norm(A) A 为矩阵，求欧几里德范数$\|A\|_2$，即 A 的最大奇异值。
- n = norm(A,1)求 A 的列范数$\|A\|_1$，即 A 的列向量的 1-范数的最大值。
- n = norm(A,2)求 A 的欧几里德范数$\|A\|_2$，与 norm(A)相同。
- n = norm(A,inf)求行范数$\|A\|_\infty$，即 A 的行向量的 1-范数的最大值，即：max(sum(abs(A')))。
- n = norm(A, 'fro') 求矩阵 A 的 Frobenius 范数 $\|A\|_F=\sqrt{\sum_i\sum_j|a_{ij}|^2}$，

即 sqrt(sum(diag(A'*A)))，不能用矩阵 p-范数的定义来求。

【功能介绍】 求矩阵的范数。

【实例 2.57】 求矩阵 *A* 的几种范数。

```
>> A=[3.3,2.2,1.5;2,4.6,3.9;3,6.4,2.9]
A =
    3.3000    2.2000    1.5000
    2.0000    4.6000    3.9000
    3.0000    6.4000    2.9000
>> n1=norm(A)
n1 =
   10.5389
>> n2=norm(A,inf)
n2 =
   12.3000
>> n3=norm(A,1)
n3 =
   13.2000
```

```
>> n4 =norm(A,2)
n4 =
   10.5389
```

【实例讲解】 上例中分别求取了欧几里德范数$\|A\|_2$，行范数$\|A\|_\infty$，1-范数和 2-范数，其中 2-范数与欧几里德范数$\|A\|_2$是相同的。

2.2.28 cond 函数——求矩阵的条件数

【语法说明】

- c = cond(X)：求 X 的 2-范数的条件数，即 X 的最大奇异值和最小奇异值的商。
- c = cond(X,p)：求 p-范数的条件数，p 的值可以是 1、2、inf 或者'fro'。

【功能介绍】 求矩阵的条件数。

【实例 2.58】 求矩阵 $A=\begin{bmatrix} 2 & 3 & 7 \\ 4 & 8 & 10 \\ 1.3 & 3.6 & 2 \end{bmatrix}$ 的条件数。

```
>> A = [2,3,7;4,8,10;1.3,3.6,2]
A =
    2.0000    3.0000    7.0000
    4.0000    8.0000   10.0000
    1.3000    3.6000    2.0000
>> x = cond(A,2)
x =
  212.7160
```

【实例讲解】 线性方程组 AX=b 的条件数是一个大于或者等于 1 的实数，用来衡量关于数据中的扰动，也就是 A/或 b 对解 X 的灵敏度。一个差条件的方程组的条件数很大。条件数的定义为：$cond(A) \quad \|A\|\|A^{-1}\|$。

2.2.29 condest 函数——1-范数的条件数估计

【语法说明】

- c = condest (A)：方阵 A 的 1-范数的条件数的下界估值。

- [c,v] = condest (A) ：v 为向量，满足$\| Av \|=\dfrac{\| A\|\cdot\| v\|}{c}$，即 norm(A*v,1) =norm(A,1)*norm(v,1)/c。
- [c,v] = condest (A,t) ：求上面的 c 和 v，同时显示出关于计算的步骤信息。如果 t=1，则计算的每步都显示出来；如果 t=-1，则给出商 c/rcond(A)。

【功能介绍】 求 1-范数的条件数估计。

【实例 2.59】 求矩阵 $\boldsymbol{A}=\begin{bmatrix}2 & 3 & 7\\ 4 & 8 & 10\\ 1.3 & 3.6 & 2\end{bmatrix}$ 的 1-范数的条件估计。

```
>> A = [2,3,7;4,8,10;1.3,3.6,2]
A =
    2.0000    3.0000    7.0000
    4.0000    8.0000   10.0000
    1.3000    3.6000    2.0000
>> c = condest(A)
c =
  240.6667
```

【实例讲解】 范数的条件数是矩阵的重要属性，可以用来设定矩阵的重要运算属性。

2.2.30 rcond 函数——矩阵可逆的条件数估值

【语法说明】 c = rcond(A)：对于差条件矩阵 A 来说，给出一个接近于 0 的数；对于好条件矩阵 A，则给出一个接近于 1 的数。

【功能介绍】 求矩阵可逆的条件数估值。

【实例 2.60】 对于矩阵 $\boldsymbol{A}=\begin{bmatrix}1 & 4 & 6\\ 3 & 5 & -3\\ -5 & 9 & 12\end{bmatrix}$ 求取其可逆的条件数估计值。

```
>> A=[1 4 6;3 5 -3;-5 9 12]

A =
     1     4     6
```

```
     3     5    -3
    -5     9    12

>> c = rcond(A)

c =

0.0938
```

【实例讲解】 得到的结果 c 更接近于 0，所以这个矩阵是一个不好的条件矩阵。

2.2.31 condeig 函数——特征值的条件数

【语法说明】

- c = condeig(A)：返回矩阵 A 的特征值的条件数。
- [V,D,c] = condeig(A)：D 为 A 的特征值对角阵，V 为 A 的特征向量。

【功能介绍】 求矩阵特征值的条件数。

【实例 2.61】 求矩阵 $\boldsymbol{A}=\begin{bmatrix}1 & 2 & 3\\ 4 & 7 & 9\\ 12 & 45 & 23.8\end{bmatrix}$ 的条件数。

```
A =
    1.0000    2.0000    3.0000
    4.0000    7.0000    9.0000
   12.0000   45.0000   23.8000
>> B = condeig(A)
B =
    1.2971
    1.1408
1.4177
```

【实例讲解】 如果用户想知道更加详细的结果，可以使用上面的第二个函数。

2.2.32 rank 函数——矩阵的秩

【语法说明】

- k = rank (A)：求矩阵 A 的秩。

- k = rank (A,tol)：tol 为给定误差。

【功能介绍】 求矩阵的秩。

【实例 2.62】 求矩阵 $\boldsymbol{A}=\begin{bmatrix}1 & 2 & 3\\4 & 7 & 9\\12 & 45 & 23.8\end{bmatrix}$ 的秩。

```
>> A=[1 2 3;4 7 9;12 45 23.8]

A =
    1.0000    2.0000    3.0000
    4.0000    7.0000    9.0000
   12.0000   45.0000   23.8000

>> k = rank (A) %求矩阵 A 的秩

k =

     3

>> k2 = rank (A,0.001)  %tol=0.001 为给定误差

k2 =

     3
```

【实例讲解】 无论是否指定误差，矩阵的秩都是 3。

2.2.33 diag 函数——矩阵对角线元素的抽取

【语法说明】

- X = diag(v,k)：以向量 v 的元素作为矩阵 X 的第 k 条对角线元素，当 k=0 时，v 为 X 的主对角线；当 k>0 时，v 为上方第 k 条对角线；当 k<0 时，v 为下方第 k 条对角线。

- X = diag(v)：以 v 为主对角线元素，其余元素为 0 构成 X。

- v = diag(X,k)：抽取 X 的第 k 条对角线元素构成向量 v。k=0：抽取主对角线元素；k>0：抽取上方第 k 条对角线元素；k<0 抽取下方第 k 条对角线元素。
- v = diag(X)：抽取主对角线元素构成向量 v。

【功能介绍】 抽取矩阵对角线元素。

【实例 2.63】 抽取矩阵的对角线元素。

```
>> v=[1 2 3];
>> x=diag(v,-1)
x =
     0     0     0     0
     1     0     0     0
     0     2     0     0
     0     0     3     0
>> A=[1 2 3;4 5 6;7 8 9]
A =
     1     2     3
     4     5     6
     7     8     9
>> v=diag(A,1)
v =
     2
     6
```

【实例讲解】 当 diag(A,k)中的 k = 1 时表示抽取以主对角线为中心的上面的一列元素，所以是 2 和 6。

2.2.34 tril 函数——下三角阵的抽取

【语法说明】

- L = tril(X)：抽取 X 的主对角线的下三角部分构成矩阵 L。
- L = tril(X,k)：抽取 X 的第 k 条对角线的下三角部分，k=0 为主对角线，k>0 为主对角线以上，k<0 为主对角线以下。

【功能介绍】 抽取矩阵的下三角矩阵。

【实例 2.64】 抽取魔方矩阵的下三角矩阵。

```
>> A=magic(4)
A =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
>> B=tril(A,1)
B =
    16     2     0     0
     5    11    10     0
     9     7     6    12
     4    14    15     1
```

【实例讲解】 没有取出数据的元素部分用 0 补齐。

2.2.35 triu 函数——上三角阵的抽取

【语法说明】

- U = triu(X)：抽取 X 的主对角线的上三角部分构成矩阵 U。
- U = triu(X,k)：抽取 X 的第 k 条对角线的上三角部分，k=0 为主对角线，k>0 为主对角线以上，k<0 为主对角线以下。

【功能介绍】 抽取矩阵的上三角矩阵。

【实例 2.65】 抽取矩阵的上三角矩阵。

```
>> A=magic(4)
A =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
>> B=triu(A,-1)
B =
    16     2     3    13
     5    11    10     8
     0     7     6    12
     0     0    15     1
```

【实例讲解】 没有取出数据的元素部分用 0 补齐。

2.2.36　reshape 函数——矩阵变维

【语法说明】

- B = reshape(A,m,n) ：返回以矩阵 A 的元素构成的 m×n 矩阵。
- B = reshape(A,m,n,p,…)：将矩阵 A 变维为 m×n×p×…
- B = reshape(A,[m n p…])：将矩阵 A 变维为 m×n×p×…
- B = reshape(A,siz)：由 siz 决定变维的大小，元素个数与 *A* 中元素个数相同。

【功能介绍】 矩阵变维。

【实例 2.66】 对矩阵 A 用 Reshape 函数进行变维操作。

```
>> a=[1:20];
>> B=reshape(a,4,5)
```

上面函数得到的结果为：

```
B =
     1     5     9    13    17
     2     6    10    14    18
     3     7    11    15    19
     4     8    12    16    20
```

【实例讲解】 1：20 表示取值的范围为 1～20。

2.2.37　rot90 函数——矩阵旋转语法说明

- B = rot90 (A)：将矩阵 A 逆时针方向旋转 90°。
- B = rot90 (A,k)：将矩阵 A 逆时针方向旋转（k×90°），k 可取正负整数。

【功能介绍】 矩阵旋转 90°或 90°的倍数。

【实例 2.67】 对矩阵 A 进行旋转操作。

```
>> A=[1 2 3;4 5 6;7 8 9]
A =
     1     2     3
     4     5     6
     7     8     9
```

```
>> Y1=rot90(A),Y2=rot90(A,-1)
Y1 =      %逆时针方向旋转
     3     6     9
     2     5     8
     1     4     7
Y2 =      %顺时针方向旋转
     7     4     1
     8     5     2
9     6     3
```

【实例讲解】 用户要注意区分旋转和矩阵转置的区别。

2.2.38 fliplr 函数——矩阵的左右翻转

【语法说明】 B = fliplr(A)：将矩阵 *A* 左右翻转。

【功能介绍】 将矩阵左右旋转。

【实例 2.68】 将矩阵 $\boldsymbol{A}=\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\end{bmatrix}$ 进行左右旋转。

```
>>  A=[1 2 3;4 5 6;7 8 9]

A =

     1     2     3
     4     5     6
     7     8     9

>> B = fliplr(A)    %将矩阵 A 左右翻转

B =

     3     2     1
     6     5     4
     9     8     7
```

【实例讲解】 从上面的结果中可以看出，实行翻转的矩阵，中间一列不会有变化。

2.2.39　flipud 函数——矩阵的上下翻转

【语法说明】 B = flipud(A)：将矩阵 A 上下翻转。

【功能介绍】 将矩阵上下翻转。

【实例 2.69】 对矩阵 *A* 做上下左右反转操作。

```
>> A=[1 2 3;4 5 6]
A =
     1     2     3
     4     5     6
>> B1=fliplr(A),B2=flipud(A)
B1 =
    3     2     1
    6     5     4
B2 =
    4     5     6
    1     2     3
```

【实例讲解】 这个函数和上面的函数类似，只是翻转的方向不同。

2.2.40　flipdim 函数——按指定维数翻转矩阵

【语法说明】 B = flipdim(A,dim)：flipdim(A,1) = flipud(A)，并且 flipdim(A,2)=fliplr(A)。

【功能介绍】 按指定维数翻转矩阵。

【实例 2.70】 对矩阵 ***A*** 分别按一维和二维翻转。

```
>> A=[1 2 3;4 5 6]
A =
     1     2     3
     4     5     6
>> B1=flipdim(A,1),B2=flipdim(A,2)
B1 =
    4     5     6
    1     2     3
B2 =
    3     2     1
    6     5     4
```

【实例讲解】 显然这个函数比前面两个函数复杂，用户可以

使用这个函数实现前面两个函数的结果。

2.2.41 repmat 函数——复制和平铺矩阵

【语法说明】

- B = repmat(A,m,n)：将矩阵 A 复制 m×n 块，即 B 由 m×n 块 A 平铺而成。
- B = repmat(A,[m n])：与上面一致
- B = repmat(A,[m n p…]) ：B 由 m×n×p×…个 A 块平铺而成。
- repmat(A,m,n)：当 A 是一个数 a 时，该函数产生一个全由 a 组成的 m×n 矩阵。

【功能介绍】 复制平铺矩阵。

【实例 2.71】 以矩阵 *A* 为基本单位产生一个 3×4 的矩阵。

```
>> A=[1 2;5 6]
A =
     1     2
     5     6
>> B=repmat(A,3,4)
B =
     1     2     1     2     1     2     1     2
     5     6     5     6     5     6     5     6
     1     2     1     2     1     2     1     2
     5     6     5     6     5     6     5     6
     1     2     1     2     1     2     1     2
     5     6     5     6     5     6     5     6
```

【实例讲解】 产生的这个大矩阵可以把 *A* 当作一个元素，然后平铺成 3 行 4 列。

2.2.42 矩阵的比较函数

【语法说明】

- >：大于关系函数。
- <：小于关系函数。
- ==：等于关系函数。

- <=：小于等于关系函数。
- >=：大于等于关系函数。
- ～ =：不等于关系函数。

【功能介绍】 矩阵的比较函数是针对于两个矩阵对应元素的，所以在使用关系运算时，首先应该保证两个矩阵的维数一致或其中一个矩阵为标量。关系运算是对两个矩阵的对应运算进行比较，若关系满足，则将结果矩阵中该位置元素置为 1，否则置 0。

【实例 2.72】 把矩阵 $\boldsymbol{A}=\begin{bmatrix}0 & 2 & 1 & 4\\0 & 7 & 7 & 2\end{bmatrix}$ 和 $\boldsymbol{B}=\begin{bmatrix}1 & 2 & 3 & 4\\5 & 6 & 7 & 8\end{bmatrix}$ 进行各种比较，并得出结论。

```
>> A=[1 2 3 4;5 6 7 8];B=[0 2 1 4;0 7 7 2];
>> C1=A==B, C2=A>=B, C3=A~=B
C1 =
     0     1     0     1
     0     0     1     0
C2 =
     1     1     1     1
     1     0     1     1
C3 =
     1     0     1     0
     1     1     0     1
```

【实例讲解】 上例中比较每个元素，如果满足对应元素的值为 1，否则为 0。

2.2.43　矩阵取整运算

【语法说明】

- floor(A)：将 A 中元素按−∞方向取整，即取不足整数。
- ceil(A)：将 A 中元素按+∞方向取整，即取过剩整数。
- round (A)：将 A 中元素按最近的整数取整，即四舍五入取整。
- fix (A)：将 A 中元素按离 0 近的方向取整。

【功能介绍】 对矩阵进行取整运算。

【实例 2.73】 先随机生成一个矩阵 *A*，对 *A* 做取整操作。

```
>> A=-2.6+3*rand(3)
A =
   -0.3854    0.2064    0.0809
   -2.0712    0.1507   -2.4263
   -1.3829   -1.3692   -1.5414
```

上面函数得到结果为：

```
>> B1=floor(A),B2=ceil(A),B3=round(A),B4=fix(A)
B1 =
    -1     0     0
    -3     0    -3
    -2    -2    -2
B2 =
     0     1     1
    -2     1    -2
    -1    -1    -1
B3 =
     0     0     0
    -2     0    -2
    -1    -1    -2
B4 =
     0     0     0
    -2     0    -2
    -1    -1    -1
```

【实例讲解】 分别对以上所述的4种取整方式进行了演示。

2.2.44 rat 函数——用有理数形式表示矩阵

【语法说明】 [n,d]=rat (A)：将A表示为两个整数矩阵相除，即A=n./d。

【功能介绍】 有理数形式表示矩阵。

【实例2.74】 对随机生成的矩阵 ***A*** 表示成有理数形式。

```
>> A = -2.6+3*rand(3)
A =
   -0.1605   -1.9917   -1.7834
   -2.5704   -2.0038   -2.0036
   -2.1833   -0.7886   -2.5542
>> [n,d]=rat(A)
```

计算结果为：

```
n =
       -116         58           17
       -640         74        -1235
       -307       -471         -242
d =

   301   281   210
   309   491   509
   222   344   157
```

【实例讲解】 得到的矩阵 ***n*** 和 ***d*** 为两个用来表示矩阵 ***A*** 的整数矩阵，二者相除的结果即为 A。

2.2.45 rem 函数——矩阵元素的余数

【语法说明】 C = rem (A, x)：表示 A 矩阵除以模数 x 后的余数。若 x=0，则定义 rem(A, 0)=NaN，若 x≠0，则整数部分由 fix(A./x)表示，余数 C=A-x.*fix (A./x)。允许模 x 为小数。

【功能介绍】 求矩阵中元素的余数。

【实例 2.75】 对矩阵 ***A*** 取余数。

```
A =
    3.0000    4.0000    5.0000
    6.0000    7.0000    8.0000
    2.6000    4.9800    5.3332
>> rem(A,1.98)
ans =
    1.0200    0.0400    1.0400
    0.0600    1.0600    0.0800
    0.6200    1.0200    1.3732
```

【实例讲解】 实际上，上面函数的功能是将矩阵中的所有元素都和设置的参数相除，然后得到的结果是除法的结果。

2.2.46 矩阵逻辑运算函数

【语法说明】

- A&B 或 and(A, B) ：A 与 B 对应元素进行与运算，若两个数均非 0，则结果元素的值为 1，否则为 0。

- A|B 或 or(A, B) ：A 与 B 对应元素进行或运算，若两个数均为 0，则结果元素的值为 0，否则为 1。
- ～A 或 not (A)：若 A 的元素为 0，则结果元素为 1，否则为 0。
- xor (A,B) ：A 与 B 对应元素进行异或运算，若相应的两个数中一个为 0，一个非 0，则结果为 0，否则为 1。

【功能介绍】 矩阵的与、或、非及异或运算。

【实例 2.76】 矩阵 ***A*** 和 ***B*** 的逻辑运算。

```
>> A=[0 2 3 4;1 3 5 0],B=[1 0 5 3;1 5 0 5]
A =
     0     2     3     4
     1     3     5     0
B =
     1     0     5     3
     1     5     0     5
>> C1=A&B,C2=A|B,C3=~A,C4=xor(A,B)
C1 =
     0     0     1     1
     1     1     0     0
C2 =
     1     1     1     1
     1     1     1     1
C3 =
     1     0     0     0
     0     0     0     1
C4 =
     1     1     0     0
     0     0     1     1
```

【实例讲解】 通过逻辑运算后，得到的结果是含有逻辑数值的矩阵。

2.2.47 符号矩阵的四则运算函数

【语法说明】 MATLAB 7.0 中把符号矩阵的四则运算简化为与数值矩阵完全相同的运算方式，其运算符为：加（+）、减（−）、乘（×）、除（/、\）等，或符号矩阵的和（symadd）、差（symsub）、乘（symmul）。

【功能介绍】 对符号矩阵进行四则运算。

【实例 2.77】 把符号矩阵 ***A*** 和 ***B*** 做相减和相除的操作。

```
>>A=sym('[1/x,x/(x+1);x/(x+2),x/(x+3)]')
>>B=sym('[x,1;x+2,'0]');
>>C=B-A
>>D=A\B
```

计算结果为：

```
C=
x-1/x  1-1/(x+1)
x+2-1/(x+2)   -1/(x+3)
D=
-6*x-2*x^3-7*x^2      1/2*x^3+x+3/2*x^2
6+2*x^3+10*x^2+14*x    -2*x^2-3/2*x-1/2*x^3
```

【实例讲解】 *C* 是多项式 *B-A* 的结果。*D* 是 *A/B* 的结果。

2.2.48 sym 函数——数值矩阵转化为符号矩阵

【语法说明】 B=sym(A)。

【功能介绍】 把数值矩阵转化为符号矩阵。

【实例 2.78】 将矩阵 $\boldsymbol{A}=\begin{bmatrix} 2/3 & sqrt(2) & 0.222 \\ 1.4 & 1/0.23 & \log(3) \end{bmatrix}$转化成符号矩阵。

```
>> A=[2/3,sqrt(2),0.222;1.4,1/0.23,log(3)]
A =
   0.6667    1.4142    0.2220
   1.4000    4.3478    1.0986
>> B=sym(A)
B =
[  2/3,      sqrt(2),                    111/500]
[   7/5,   100/23,   4947709893870346*2^(-52)]
```

【实例讲解】 得到的结果与 ***A*** 矩阵相同，数值元素变成了分数或者符号元素，但是表示的数值大小相同。

2.2.49 factor 函数——符号矩阵的因式分解

【语法说明】 factor(s) ：符号表达式 s 的因式分解函数。s 为符号矩阵或符号表达式，常用于多项式的因式分解。

【功能介绍】 对符号矩阵进行因式分解。

【实例 2.79】 将 x^9-1 分解因式。

```
>> syms x
factor(x^9-1)
```

结果显示为：

```
ans =
(x-1)*(x^2+x+1)*(x^6+x^3+1)
```

【实例讲解】 最后得到的结果为分解后的结果。

【实例2.80】 问“λ”取何值时，齐次方程组$\begin{cases}(1-\lambda)x_1-2x_2+4x_3=0\\2x_1+(3-\lambda)x_2+x_3=0\\x_1+x_2+(1-\lambda)x_3=0\end{cases}$有非 0 解？

在 Matlab 编辑器中建立 M 文件：

```
syms k
A=[1-k -2 4;2 3-k 1;1 1 1-k];
D=det(A)
factor(D)
```

其结果显示如下：

```
D =
-6*k+5*k^2-k^3
ans =
-k*(k-2)*(-3+k)
```

从而得到：当 k=0、k=2 或 k=3 时，原方程组有非 0 解。

【实例讲解】 并不是所有的数学表达式都可以进行因式分解。对于不能进行因式分解的表达式，函数返回原来的结果。

2.2.50 expand 函数——符号矩阵的展开

【语法说明】 expand(s)：符号表达式 s 的展开函数。s 为符号矩阵或表达式。常用在多项式的因式分解中，也常用于三角函数、指数函数和对数函数的展开中。

【功能介绍】 对符号矩阵进行展开。

【实例 2.81】 将$(x+1)^3$、$\sin(x+y)$展开。

在 Matlab 编辑器中建立 M 文件：

```
syms  x  y
p=expand((x+1)^3)
q=expand(sin(x+y))
```

则结果显示为：

```
p =
    x^3+3*x^2+3*x+1
q =
sin(x)*cos(y)+cos(x)*sin(y)
```

【实例讲解】 p、q 分别为符号矩阵展开的结果。

2.2.51 simple 或 simplify 函数——符号简化

【语法说明】

- simple (s)： s 是矩阵或表达式。
- [R,how]=simple (s) ：R 为返回的最简形，how 为简化过程中使用的主要方法。Simple(s)将表达式 s 的长度简化到最短。若还想让表达式更加精美，可使用函数 Pretty。

【功能介绍】 简化符号矩阵。

【实例 2.82】 计算行列式 $\begin{vmatrix} 1 & 1 & 1 & 1 \\ a & b & c & d \\ a^2 & b^2 & c^2 & d^2 \\ a^4 & b^4 & c^4 & d^4 \end{vmatrix}$ 的值。

在 Matlab 编辑器中建立 M 文件：

```
syms  a  b  c  d
A=[1 1 1 1;a b c d;a^2 b^2 c^2 d^2;a^4 b^4 c^4 d^4];
d1=det(A)
d2=simple(d1)     %化简表达式 d1
```

计算结果如下：

```
d1 =
b*c^2*d^4-b*d^2*c^4-b^2*c*d^4+b^2*d*c^4+b^4*c*d^2-b^
4*d*c^2-a*c^2*d^4+a*d^2*c^4+a*b^2*d^4-a*b^2*c^4-a*b^4*d^
2+a*b^4*c^2+a^2*c*d^4-a^2*d*c^4-a^2*b*d^4+a^2*b*c^4+a^2*
b^4*d-a^2*b^4*c-a^4*c*d^2+a^4*d*c^2+a^4*b*d^2-a^4*b*c^2-
a^4*b^2*d+a^4*b^2*c
d2 =
```

```
(-d+c)*(b-d)*(b-c)*(-d+a)*(a-c)*(a-b)*(a+c+d+b)
(-d+c)(b-d)(b-c)(-d+a)(a-c)(a-b)(a+c+d+b)
```

【实例讲解】 从上面的结果可以看出，simple 函数可以让结果中的同类项合并，从而简化结果。

2.2.52 numel 函数——确定矩阵元素个数

【语法说明】 n = numel(a)：计算矩阵 ***A*** 中元素的个数。

【功能介绍】 求出矩阵中元素的总数。

【实例 2.83】 计算矩阵 ***A*** 中元素的个数。

```
>> A = [1,2,3,4;5,6,7,8;9,12,11,14;2,4,6,9]
A =
     1     2     3     4
     5     6     7     8
     9    12    11    14
     2     4     6     9
>> numel(A)
ans =
16
```

【实例讲解】 在矩阵 ***A*** 中共有 16 个元素，所以得到的结果是 16。

2.3 矩阵分解

在数值运算中，矩阵除了做一些简单的操作外，解方程组是其更主要的应用。通常用矩阵的变换去解方程组是非常麻烦的，影响速度也会影响精度。所以在解方程组之前对方程组的系数矩阵进行分解是十分必要的。下面的小节中将要讲解矩阵分解的相关内容。

2.3.1 chol 函数——Cholesky 分解

【语法说明】

- R = chol(X)：如果 X 为 n 阶对称正定矩阵，则存在一个奇异上三角阵 R，满足 R'*R = X；若 X 非正定，则产生错误警告信息。

□ [R,p] = chol(X)：不产生任何错误信息，若 X 为正定阵，则 p=0，R 与上相同；若 X 非正定，则 p 为正整数，R 是有序的上三角阵。

【功能介绍】 实现 Cholesky 分解。

【实例 2.84】 首先建立一个 pascal 矩阵，对这个矩阵进行 Cholesky 分解。

```
>> X=pascal(4)     %产生 4 阶 pascal 矩阵
X =
     1     1     1     1
     1     2     3     4
     1     3     6    10
     1     4    10    20
>> [R,p]=chol(X)
R =
     1     1     1     1
     0     1     2     3
     0     0     1     3
     0     0     0     1
p =
     0
```

【实例讲解】 在矩阵 A 中共有 16 个元素，所以得到的结果是 16。

2.3.2 lu 函数——LU 分解

【语法说明】

□ [L,U] = lu(X) ：U 为上三角阵，L 为下三角阵或其变换形式，满足 LU=X。

□ [L,U,P] = lu(X) ：U 为上三角阵，L 为下三角阵，P 为单位矩阵的行变换矩阵，满足 LU=PX。

【功能介绍】 实现矩阵的 LU 分解。矩阵的三角分解又称 LU 分解，它是将一个矩阵分解成一个下三角矩阵 L 和一个上三角矩阵 U 的乘积，即 A=LU。从而方便后面的方程（方程组）求解。

【实例 2.85】 把矩阵 $A=\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\end{bmatrix}$ 做 LU 分解。

```
>> A=[1 2 3;4 5 6;7 8 9];
>> [L,U]=lu(A)
```

```
L =
   0.1429    1.0000         0
   0.5714    0.5000    1.0000
   1.0000         0         0
U =
   7.0000    8.0000    9.0000
        0    0.8571    1.7143
        0         0    0.0000
>> [L,U,P]=lu(A)
L =
1.0000         0         0
   0.1429    1.0000         0
   0.5714    0.5000    1.0000
U =
   7.0000    8.0000    9.0000
        0    0.8571    1.7143
        0         0    0.0000
P =
     0     0     1
     1     0     0
     0     1     0
```

【实例讲解】 这个例子中第一个函数求得的 L、U 为分解后的结果，而第二个是经过了行变换之后的结果，所以，方程转化为 LU=PX。

2.3.3 qr 函数——QR 分解

【语法说明】

- [Q,R] = qr(A)：求得正交矩阵 Q 和上三角阵 R，Q 和 R 满足 A=QR。
- [Q,R,E] = qr(A)：求得正交矩阵 Q 和上三角阵 R，E 为单位矩阵的变换形式，R 的对角线元素按大小降序排列，满足 AE=QR。
- [Q,R] = qr(A,0)：产生矩阵 A 的“经济大小”分解。
- [Q,R,E] = qr(A,0)：E 的作用是使得 R 的对角线元素降序，且 Q*R=A*E。
- R = qr(A)：稀疏矩阵 A 的分解，只产生一个上三角阵 R，满足 R'*R = A'*A，这种方法计算 A'×A 时减少了内在数字信息的损耗。

- [C,R] = qr(A,b)：用于稀疏最小二乘问题。minimize||Ax-b|| 的两步解为[C,R] = qr(A,b)，x = R\c。
- R = qr(A,0)：针对稀疏矩阵 A 的经济型分解。
- [C,R] = qr(A,b,0)：针对稀疏最小二乘问题的经济型分解。

【功能介绍】 将矩阵 A 分解成一个正交矩阵与一个上三角矩阵的乘积。

【实例 2.86】 把矩阵 $A=\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\\10 & 11 & 12\end{bmatrix}$ 做 QR 分解。

```
>>A =[ 1  2  3;4  5  6;  7  8  9;  10  11  12];
>>[Q,R] = qr(A)
Q =
   -0.0776   -0.8331    0.5444    0.0605
   -0.3105   -0.4512   -0.7709    0.3251
   -0.5433   -0.0694   -0.0913   -0.8317
   -0.7762    0.3124    0.3178    0.4461
R =
  -12.8841   -14.5916   -16.2992
         0    -1.0413    -2.0826
         0          0     0.0000
         0          0          0
```

【实例讲解】 在线性代数中，QR 分解是一个很重要的内容。在实际中，进行 QR 分解需要很复杂的运算。

2.3.4　qrdelete 函数——从 QR 分解中删除列

【语法说明】 [Q,R]= qrdelete(Q,R,j)：返回将矩阵 A 的第 j 列移去后的新矩阵的 QR 分解。

【功能介绍】 矩阵 A 去除一列后再进行 QR 分解。

【实例 2.87】 以矩阵 $A=\begin{bmatrix}-12 & -34 & -45\\123 & 34 & 20\\23.8 & 6 & 42\end{bmatrix}$ 为例进行 QR 分解。

```
>> A=[-12,-34,-45;123,34,20;23.8,6,42]
A =
```

```
  -12.0000  -34.0000  -45.0000
  123.0000   34.0000   20.0000
   23.8000    6.0000   42.0000
>> [Q,R]=qr(A)
Q =
   -0.0953   -0.9953   -0.0185
    0.9773   -0.0901   -0.1917
    0.1891   -0.0364    0.9813
R =
  125.8548   37.6052   31.7795
         0   30.5589   41.4585
         0         0   38.2133
>> [Q,R]=qrdelete(Q,R,2)
Q =
   -0.0953   -0.7444    0.6609
    0.9773   -0.1961   -0.0799
    0.1891    0.6383    0.7462
R =
  125.8548   31.7795
         0   56.3832
         0         0
```

【实例讲解】 用户可以将这个函数和 QR 函数进行比较分析。

2.3.5 qinsert 函数——从 QR 分解中添加列

【语法说明】 [Q,R] = qrinsert(Q,R,j,x)：在矩阵 A 中第 j 列插入向量 x 后的新矩阵进行 QR 分解。若 j 大于 A 的列数，表示在 A 的最后插入列 x。

【功能介绍】 在原矩阵中添加一列后再进行分解。

【实例 2.88】 对矩阵 $A=\begin{bmatrix}-14 & -50 & -15\\ 87 & 187 & 346\\ -27 & -9 & -55\end{bmatrix}$ 中插入向量 x=[35 10 7]后进行 QR 分解。

```
>> A=[-14 -50 -15;87 187 346;-27 -9 -55];
>> x=[35 10 7]';
>> [Q,R]=qrinsert(Q,R,4,x)
Q =
```

```
   -0.2671   -0.7088    0.6529
    0.9625   -0.1621    0.2176
   -0.0484    0.6865    0.7255
R =
  557.9418  187.0321  567.8424   -0.0609
         0    0.0741    3.4577  -21.6229
         0         0    0.1451   30.1073
```

【实例讲解】 这个函数和前面小节的 qrdelete 函数操作正好相反。

2.3.6 schur 函数——Schur 分解

【语法说明】

- T = schur(A)：产生 schur 矩阵 T，即 T 的主对角线元素为特征值的三角阵。
- T = schur(A,flag)：flag='complex'，显示复特征根；flag='real'，显示实特征根。
- [U,T] = schur(A,…)：返回正交矩阵 U 和 schur 矩阵 T，满足 A = U*T*U'。

【功能介绍】 对矩阵进行 Schur 分解。

【实例 2.89】 对指定矩阵 $\boldsymbol{A}=\begin{bmatrix}-14 & -50 & -15\\ 53 & 180 & 56\\ -27 & -9 & -25\end{bmatrix}$ 做 Schur 分解。

```
>>A = [ -14  -50  -15; 53  180  56; -27  -9  -25 ]
A =
   -14   -50   -15
    53   180    56
   -27    -9   -25
>> T = schur(A)
T =
  160.0263  -19.0234  364.3823
         0  -21.0132   96.0387
         0  -12.5966  -21.0132
>> T = schur(A,'complex')
T =
  1.0e+002 *
   1.6003              -1.2408 - 0.1789i  -0.0648 - 3.4261i
```

```
          0          -0.2101 + 0.3478i  -0.8344 - 0.0000i
          0                  0          -0.2101 - 0.3478i
>> T = schur(A,'real')
T =
  160.0263  -19.0234  364.3823
         0  -21.0132   96.0387
         0  -12.5966  -21.0132
>> [U,T]=schur(H)
U =
   -0.2688    0.6428    0.7173
    0.9632    0.1736    0.2053
   -0.0074   -0.7461    0.6658
T =
  164.7789   24.4181  116.5465
         0    0.1146   28.2606
         0         0  -23.8935
```

【实例讲解】 按照语法说明中的讲解顺序分别对矩阵进行操作，可以看到当 flag 分别为 complex 和 real 时的分解情况。

2.3.7 rsf2csf 函数——实 Schur 向复 Schur 转化

【语法说明】 [U,T] = rsf2csf (U,T)将实舒尔形式转化成复舒尔形式。

【功能介绍】 实舒尔形式向复舒尔形式转化。

【实例 2.90】 将矩阵 $\boldsymbol{A}=\begin{bmatrix}1 & 1 & 1 & 3\\1 & 2 & 1 & 1\\1 & 1 & 3 & 1\\-2 & 1 & 1 & 4\end{bmatrix}$进行转化。

```
>> A=[1 1 1 3;1 2 1 1;1 1 3 1;-2 1 1 4];
>> [u,t]=schur (A)
u =
   -0.4916   -0.4900   -0.6331   -0.3428
   -0.4980    0.2403   -0.2325    0.8001
   -0.6751    0.4288    0.4230   -0.4260
   -0.2337   -0.7200    0.6052    0.2466
t =
    4.8121    1.1972   -2.2273   -1.0067
         0    1.9202   -3.0485   -1.8381
         0    0.7129    1.9202    0.2566
```

```
        0     0           0            1.3474
   >> [U,T]=rsf2csf (u,t)
   U =
     -0.4916  -0.2756 - 0.4411i  0.2133 + 0.5699i  -0.3428
     -0.4980  -0.1012 + 0.2163i  -0.1046 + 0.2093i  0.8001
     -0.6751  0.1842 + 0.3860i  -0.1867 - 0.3808i  -0.4260
     -0.2337  0.2635 - 0.6481i  0.3134 - 0.5448i   0.2466
   T =
      4.8121 -0.9697+1.0778i     -0.5212+2.0051i     -1.0067
      0      1.9202 + 1.4742i    2.3355               0.1117+
1.6547i
      0      0                   1.9202- 1.4742i      0.8002
+ 0.2310i
      0      0                   0                    1.3474
```

【实例讲解】 关于舒尔形式的具体知识，用户可以参考相应的书籍。

2.3.8　eig 函数——特征值分解

【语法说明】

- d = eig(A)：求矩阵 A 的特征值 d，以向量形式存放 d。
- d = eig(A,B)：A、B 为方阵，求广义特征值 d，以向量形式存放 d。
- [V,D] = eig(A)：计算 A 的特征值对角阵 D 和特征向量 V，使 AV=VD 成立。
- [V,D] = eig(A,'nobalance')：当矩阵 A 中有与截断误差数量级相差不远的值时，该指令可能更精确。
- [V,D] = eig(A,B)：计算广义特征值向量阵 V 和广义特征值阵 D，满足 AV=BVD。

【功能介绍】 求矩阵的特征值。

【实例 2.91】 求矩阵 $A=\begin{bmatrix}1 & 2 & 3\\ 40 & 34 & 56\\ -7 & 19 & -87\end{bmatrix}$ 的特征值。

```
>> A=[1,2,3;40,34,56;-7,19,-87]
A =
```

```
     1     2     3
    40    34    56
    -7    19   -87
>> d = eig(A)
d =
   44.1890
   -1.2268
  -94.9622
>> [V,D] = eig(A)
V =
    0.0555    0.7871   -0.0206
    0.9886   -0.5855   -0.3928
    0.1402   -0.1939    0.9194
D =
   44.1890         0         0
        0   -1.2268         0
        0         0  -94.9622
>> B=[2,4,7;4,5,8;9,10,23]
B =
     2     4     7
     4     5     8
     9    10    23
>> [V,D] = eig(A,B)
V =
    1.0000   -0.1704   -1.0000
    0.1436    1.0000    0.8165
   -0.3658   -0.5247    0.2347
D =
   13.6172         0         0
        0  -18.2769         0
        0         0    0.4597
```

【实例讲解】 d为特征值，V、D分别为特征对角阵和特征向量。

【实例 2.92】 求矩阵 $\boldsymbol{A}=\begin{bmatrix}0.5 & 0.25\\0.25 & 0.5\end{bmatrix}$ 的特征值。

```
>> A = [0.5 0.25; 0.25 0.5];
>> [Q,d] = eig(A)
Q =
0.7071 0.7071
-0.7071 0.7071
d =                % 对角线上的元素为特征值
```

```
0.2500 0
0 0.7500
>> Q*Q' % Q*Q'=I
ans=
1 0
0 1
>> A*Q(:,1); 0.25* Q(:,1)
ans =                          % 是A*X 的值
0.1768
-0.1768
ans =                          % 是λX 的值
0.1768
-0.1768
```

【实例讲解】 用户可以自行验证关于上面矩阵特征值的结果，这里就不重复了。

2.3.9 svd 函数——奇异值分解

【语法说明】

- s = svd (X)：返回矩阵 X 的奇异值向量。
- [U,S,V] = svd (X)：返回一个与 *X* 同大小的对角矩阵 S，两个友矩阵 U 和 V，并且满足= U*S*V'。若 A 为 m×n 阵，则 U 为 m×m 阵，V 为 n×n 阵。
- [U,S,V] = svd (X,0)：得到一个有效的分解，只计算出矩阵 U 的前 n 列，矩阵 S 的大小为 *n*×*n*。

【功能介绍】 求矩阵的奇异值。

【实例 2.93】 求矩阵 $A=\begin{bmatrix}1 & 2\\3 & 4\\5 & 6\\7 & 8\end{bmatrix}$ 的矩阵的奇异值向量。

```
>> A=[1 2;3 4;5 6;7 8];
A =
     1     2
     3     4
```

```
     5     6
     7     8
>> [U,S,V]=svd(A)
U =
   -0.1525   -0.8226   -0.3945   -0.3800
   -0.3499   -0.4214    0.2428    0.8007
   -0.5474   -0.0201    0.6979   -0.4614
   -0.7448    0.3812   -0.5462    0.0407
S =
   14.2691         0
         0    0.6268
         0         0
         0         0
V =
   -0.6414    0.7672
   -0.7672   -0.6414
>> [U,S,V]=svd(A,0)
U =
   -0.1525   -0.8226
   -0.3499   -0.4214
   -0.5474   -0.0201
   -0.7448    0.3812
S =
   14.2691         0
         0    0.6268
V =
   -0.6414    0.7672
   -0.7672   -0.6414
```

【实例讲解】 用户可以自行比较上面的两个函数。

2.3.10 gsvd 函数——广义奇异值分解

【语法说明】

- [U,V,X,C,S] = gsvd(A,B)：返回友矩阵 U 和 V、普通方阵 X、非负对角矩阵 C 和 S，并且满足 A = U*C*X'，B = V*S*X'，$C'*C+S'*S$ = I （I 为单位矩阵）；A 和 B 的列数必须相同。
- [U,V,X,C,S] = gsvd(A,B,0)：与上面相同。

- sigma = gsvd (A,B)：返回广义奇异值 sigma。

【功能介绍】 求矩阵的广义奇异值。

【实例 2.94】 先自动生成两个矩阵，而后对其进行操作。

```
>> A=reshape(1:12,3,4)       %产生 3 行 4 列矩阵，元素由
1,2,…,12 构成
A =
    1     4     7    10
    2     5     8    11
    3     6     9    12
>> B=magic(4)    %产生 4 阶魔方阵
B =
   16     2     3    13
    5    11    10     8
    9     7     6    12
    4    14    15     1
>> [U,V,X,C,S]=gsvd(A,B)
U =
    0.4082    0.7071    0.5774
   -0.8165    0.0000    0.5774
    0.4082   -0.7071    0.5774
V =
    0.2607   -0.7950   -0.5000    0.2236
   -0.4029    0.3710   -0.5000    0.6708
   -0.5452   -0.0530   -0.5000   -0.6708
    0.6874    0.4770   -0.5000   -0.2236
X =
         0   -9.4340  -17.0587    3.4641
    1.8962    8.7980  -17.0587    8.6603
    3.7924    8.1620  -17.0587   13.8564
   -5.6885   -7.5260  -17.0587   19.0526
C =
         0    0.0000         0         0
         0         0    0.0829         0
         0         0         0    1.0000
S =
    1.0000         0         0         0
         0    1.0000         0         0
         0         0    0.9966         0
         0         0         0    0.0000
```

【实例讲解】 用户可以根据前面的定义和上面结果，进行验证。

2.3.11 qz 函数——特征值问题的 QZ 分解

【语法说明】

- [AA,BB,Q,Z,V] = qz(A,B)：A、B 为方阵，产生上三角阵 AA 和 BB，正交矩阵 Q、Z 或其列变换形式，V 为特征向量阵。
- [AA,BB,Q,Z,V] = qz(A,B,flag)：产生由 flag 决定的分解结果，flag 取值为 complex 表示复数分解（默认），取值为 real 表示实数分解。

【功能介绍】 对矩阵进行 QZ 分解。

【实例 2.95】 先自动生成两个矩阵，而后对其进行操作。

```
>> A=reshape(1:16,4,4)      %产生 3 行 4 列矩阵，元素由 1,2,…,16 构成
A =
     1     5     9    13
     2     6    10    14
     3     7    11    15
     4     8    12    16
>> B=magic(4)
B =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
>> [AA,BB,Q,Z,V] = qz(A,B)
AA =
  -10.7331   11.4742  -34.0000   -8.5524
         0    0.0000   -3.3920   -0.0000
         0         0    2.9145   -0.0000
         0         0         0   -0.0000
BB =
    0.0000   -0.0000  -34.0000    0.0000
         0   11.2758   -0.0000   12.7483
         0         0         0   -2.7696
         0         0         0    6.5320
Q =
   -0.5000   -0.5000   -0.5000   -0.5000
```

```
    0.7950   -0.3710    0.0530   -0.4770
   -0.1041   -0.7749    0.4048    0.4742
   -0.3273    0.1091    0.7638   -0.5455
Z =
   -0.2236    0.8367    0.5000         0
   -0.6708   -0.4781    0.5000   -0.2673
    0.6708   -0.1195    0.5000   -0.5345
    0.2236   -0.2390    0.5000    0.8018
V =
   -0.3333    0.5000   -0.3333    0.2527
   -1.0000   -1.0000   -1.0000    0.1210
    1.0000    0.5000    1.0000   -1.0000
    0.3333   -0.0000    0.3333    0.6263
```

【实例讲解】 用户可以根据函数运算的结果和 QZ 定义，来检测结果。

2.3.12　hess 函数——海森伯格形式的分解

【语法说明】

- H = hess(A)：返回矩阵 A 的海森伯格形式。
- [P,H] = hess(A)：P 为友矩阵，满足 A = PHP' 且 P'P = eye(size(A))。

【功能介绍】 对矩阵进行海森伯格分解。

【实例 2.96】 把矩阵 $\boldsymbol{A}=\begin{bmatrix}-3 & 5 & -53\\-38 & 8 & 20\\7 & -86 & 25\end{bmatrix}$ 做海森伯格形式的分解。

```
>>  A=[-3 5 -53;-38 8 20;7 -86 25]
A =
    -3     5   -53
   -38     8    20
     7   -86    25
>> [P,H]=hess(A)
P =
    1.0000         0         0
         0   -0.9835    0.1812
         0    0.1812    0.9835
```

```
H =
  -3.0000  -14.5189  -51.2172
  38.6394   20.3168  -19.1373
        0   86.8627   12.6832
```

【实例讲解】 如果矩阵 H 的第一子对角线下元素都是 0，则 H 为海森伯格（Hessenberg）矩阵。如果矩阵是对称矩阵，则它的海森伯格形式是对角三角阵。

2.4 线性方程的组的求解

在数值计算和工程应用中，通常将线性方程的求解分为两类：一类是方程组求特解，另一类是方程组求通解。可以通过系数矩阵的秩来判断方程组是否有解：

- 若系数矩阵的秩 $r=n$（n 为方程组中未知变量的个数），则有唯一解；
- 若系数矩阵的秩 $r<n$，则可能有无穷解；
- 线性方程组的无穷解 = 对应齐次方程组的通解+非齐次方程组的一个特解；其特解的求法属于解的第一类问题，通解部分属第二类问题。

2.4.1 直接法求线性方程组的特解

【语法说明】 直接法：利用矩阵除法求线性方程组的特解。

若方程为：$AX=b$

则求得的根为：$X=A\backslash b$

【功能介绍】 求线性方程组的特解。

【实例 2.97】 求方程组 $\begin{cases} 7x_1+x_2-3x_3=12 \\ 3x_1-x_2-3x_3=4 \\ x_1+5x_2-9x_3=6 \end{cases}$ 的特解。

解：

```
>> A=[7,1,-3;3,-1,-3;1,5,-9]
A =
     7     1    -3
     3    -1    -3
     1     5    -9
>> b=[12,4,6]'
b =
    12
     4
     6
>> R_A=rank(A)         %求秩,用来判断是否有解,有何种解
R_A =
     3
>>  X=A\b
X =
    1.5833
    0.8333
   -0.0278
```

用函数 rref 求解：

```
>> C=[A,b]    %由系数矩阵和常数列构成增广矩阵 C
C =
     7     1    -3    12
     3    -1    -3     4
     1     5    -9     6
>> R=rref(C)    %将 C 化成行最简行
R =
    1.0000         0         0    1.5833
         0    1.0000         0    0.8333
         0         0    1.0000   -0.0278
```

【实例讲解】 求解的过程中，最好首先求出秩，用来判断根的情况。如果用 rref 函数求解的话，最后一列元素为方程的特解。

【实例 2.98】 求方程组 $\begin{cases} x_1+x_2-3x_3-x_4=1 \\ 3x_1-x_2-3x_3+4x_4=4 \\ x_1+5x_2-9x_3-8x_4=0 \end{cases}$ 的一个特解。

```
>> A=[1 1 -3 -1;3 -1 -3 4;1 5 -9 -8]
A =
     1     1    -3    -1
```

```
     3     -1     -3      4
     1      5     -9     -8
>> B=[1  4  0]'
B =
     1
     4
     0
>> R_A=rank(A)    %求秩
R_A =
     2
>> X=A\B
Warning: Rank deficient, rank = 2,  tol =   8.8373e-015.
%存在误差
X =
         0
         0
   -0.5333
0.6000
```

若用rref求解，则比较精确：

```
>> A=[1 1 -3 -1;3 -1 -3 4;1 5 -9 -8]
A =
     1     1    -3    -1
     3    -1    -3     4
     1     5    -9    -8
>> B=[1  4  0]'
B =
     1
     4
     0
>> C=[A,B];     %构成增广矩阵
>> R=rref(C)
R =
    1.0000         0   -1.5000    0.7500    1.2500
         0    1.0000   -1.5000   -1.7500   -0.2500
         0         0         0         0         0
>> A=[1 1 -3 -1;3 -1 -3 4;1 5 -9 -8];
B=[1  4  0]';
>> C=[A,B];     %构成增广矩阵
>> R=rref(C)
R =
    1.0000         0   -1.5000    0.7500    1.2500
```

```
        0    1.0000   -1.5000   -1.7500   -0.2500
        0         0         0         0         0
```

【实例讲解】 由此得到解向量 X=[1.2500 – 0.2500 0 0]为方程的一个特解。

2.4.2 用矩阵的 LU 分解求方程组的解

【语法说明】 LU 分解又称 Gauss 消去分解，可把任意方阵分解为下三角矩阵的基本变换形式（行交换）和上三角矩阵的乘积。即 ***A=LU***，***L*** 为下三角阵，***U*** 为上三角阵。

则：***A*X=b*** 变成 ***L*U*X=b***，所以方程组的解 ***X=U\(L\b)***。

【功能介绍】 利用 LU 分解法求解方程组的解。

【实例 2.99】 求方程组 $\begin{cases} 4x_1+2x_2-x_3=2 \\ 3x_1-x_2+2x_3=10 \\ 11x_1+3x_2+9x_3=8 \end{cases}$ 的一个特解。

解：

```
>> A =[4 2 -1;3 -1 2;11 3 9]
A =
     4     2    -1
     3    -1     2
    11     3     9
>> C=det(A)
C =
   -90
>> b=[2 10 8]'
b =
     2
    10
     8
>> [L,U]=lu(A)
L =
    0.3636   -0.5000    1.0000
    0.2727    1.0000         0
    1.0000         0         0
U =
```

```
    11.0000    3.0000    9.0000
          0   -1.8182   -0.4545
          0         0   -4.5000
>> X=U\(L\b)
X =

    2.4000
   -4.1333
   -0.6667
```

【实例讲解】 如果方程组系数的行列式为零，那么 MATLAB 会给出警告。

2.4.3 QR 分解求方程组的解

【语法说明】 对于任何长方矩阵 ***A***，都可以进行 QR 分解，其中 ***Q*** 为正交矩阵，***R*** 为上三角矩阵的初等变换形式为：***A***=***QR***。方程 ***A****X*=*b* 变成 ***QR****X*=*b*，所以方程的解 *X*=***R***\(***Q****b*)。

【功能介绍】 利用 QR 分解解方程组。

【实例 2.100】 利用 QR 分解解上节中的方程组。

```
>> A =[4 2 -1;3 -1 2;11 3 9]
A =
     4     2    -1
     3    -1     2
    11     3     9
>> C=det(A)
C =
   -90
>> b=[2 10 8]'
b =
     2
    10
     8
>> [Q,R]=qr(A)
Q =
   -0.3310    0.4730   -0.8165
   -0.2483   -0.8785   -0.4082
   -0.9104    0.0676    0.4082
R =
```

```
  -12.0830   -3.1449   -8.3588
         0    2.0272   -1.6218
         0         0    3.6742
>> X=R\(Q\b)
X =
    2.4000
   -4.1333
   -0.6667
```

【实例讲解】 用户也可以运用正常的矩阵运算方法，来求解这个结果。

2.4.4　null 函数——求线性齐次方程组的通解

【语法说明】

- z=null：z 的列向量为方程组的正交规范基，'满足 $z' \times z = I$ 。
- $z = \text{null}(A, 'r')$：z 的列向量是方程 AX=0 的有理基。

【功能介绍】 在 Matlab 中，函数 null 用来求零空间，即满足 A・X=0 的解空间，实际上是求出解空间的一组基。

【实例 2.101】 求解方程组的通解：$\begin{cases} x_1 + 2x_2 + 2x_3 + x_4 = 0 \\ 2x_1 + x_2 - 2x_3 - 2x_4 = 0 \\ x_1 - x_2 - 4x_3 - 3x_4 = 0 \end{cases}$

解:

```
>>A=[1  2  2  1;2  1  -2  -2;1  -1  -4  -3];
>>B=null(A,'r')     %求解空间的有理基
```

运行后显示结果如下：

```
B =
      2            5/3
      -2           -4/3
      1            0
      0            1
```

或通过行最简行得到基：

```
>> B=rref(A)
B =
    1.0000         0   -2.0000   -1.6667
```

```
        0    1.0000    2.0000    1.3333
        0         0         0         0
```

写出通解：

```
syms  k1  k2
X=k1*B(:,1)+k2*B(:,2)       %写出方程组的通解
```

运行后结果如下：

```
X =
[  2*k1+5/3*k2]
[ -2*k1-4/3*k2]
[           k1]
[           k2]
%下面是其简化形式
    [2k1 + 5/3k2 ]
    [            ]
    [-2k1 - 4/3k2]
    [            ]
    [     k1     ]
    [            ]
    [     k2     ]
```

【实例讲解】 在前面的步骤中，syms 函数是将 *k*1 和 *k*2 定义为符号变量。关于这个函数的具体用法，请用户参考后面的具体章节。

2.4.5 求非齐次线性方程组的通解

求解非齐次线性方程组首先判断方程组是否有解，如果有解，再对其求通解。

通常情况下的求解步骤是：

- 判断 $AX=b$ 是否有解，若有解则进行第二步；
- 求 $AX=b$ 的一个特解；
- 求 $AX=0$ 的通解；
- $AX=b$ 的通解为 $AX=0$ 的通解和 $AX=b$ 的一个特解。

【实例 2.102】 求解方程组 $\begin{cases} x_1-2x_2+3x_3-4x_4=2 \\ 3x_1+5x_2+5x_3-8x_4=3 \\ 5x_1+6x_2+10x_3-3x_4=0 \end{cases}$

在 Matlab 中建立 M 文件如下：

```
A=[1  -2  3  -4;3  5  5  -8;5  6  10  -3];
b=[2  3  0]';
B=[A b];
n=4;
R_A=rank(A)
R_B=rank(B)
if R_A==R_B&R_A==n                %判断有唯一解
   X=A\b
elseif R_A==R_B&R_A<n             %判断有无穷解
   X=A\b                          %求特解
   C=null(A,'r')                  %求 AX=0 的基础解系
else X='该方程组无解'              %判断无解
end
```

运行后结果显示：

```
R_A =
     3
R_B =
     3
X =
       0
    -37/309
     -8/103
   -154/309
C =
    103/3
    -16/3
    -41/3
      1
```

【实例讲解】 因为系数矩阵的秩与[A b]矩阵的秩相同，所以这个方程组有解。

【实例 2.103】 求解方程组的解：$\begin{cases} 2x_1+3x_2-3x_3-6x_4=8 \\ 3x_1-4x_2-8x_3+4x_4=4 \\ x_1+5x_2-9x_3-8x_4=13 \end{cases}$

在 Matlab 编辑器中建立 M 文件如下：

```
A=[2  3  -3  -6;3  -4  -8  4;1  5  -9  -8];
b=[8 4 13]';
B=[A b];
```

```
n=4;
R_A=rank(A)
R_B=rank(B)
if R_A==R_B&R_A==n
   X=A\b
elseif R_A==R_B&R_A<n
   X=A\b
   C=null(A,'r')
else X='方程组无解'
end
```

运行后结果显示为：

```
R_A =
     3
R_B =
     3
X =
    14/23
     0
   -77/115
  -183/230
C =
    15/19
   115/76
     3/76
     1
```

【实例讲解】 通过判断秩的值，就可以知道解的情况。

2.4.6 symmlq 函数——线性方程组的 LQ 解法

【语法说明】

- x = symmlq(A,b)：求线性方程组 AX=b 的解 X。A 必须为 n 阶对称方阵，b 为 n 元列向量。A 可以是由 afun 定义并返回 A*X 的函数。如果收敛，将显示结果信息；如果收敛失败，将给出警告信息并显示相对残差 norm(b−A*x)/norm(b)和计算终止的迭代次数。
- symmlq(A,b,tol)：指定误差 tol，默认值是 1e−6。
- symmlq(A,b,tol,maxit)：maxit 指定最大迭代次数。
- symmlq(A,b,tol,maxit,M)：M 为用于对称正定矩阵的预处理因子。

- symmlq(A,b,tol,maxit,M1,M2) ：M=M1×M2。
- symmlq(A,b,tol,maxit,M1,M2,x0) ：x0 为初始估计值，默认值为 0。
- [x,flag] = symmlq(A,b,···) ：flag 的取值为 0 表示在指定迭代次数之内按要求精度收敛，为 1 表示在指定迭代次数内不收敛，为 2 表示 M 为坏条件的预处理因子，为 3 表示两次连续迭代完全相同，为 4 表示标量参数太小或太大，为 5 表示预处理因子不是对称正定的。
- [x,flag,relres] = symmlq(A,b,···)：relres 表示相对误差 norm(b−A*x)/norm(b)。
- [x,flag,relres,iter] = symmlq(A,b,···)：iter 表示计算 x 的迭代次数。
- [x,flag,relres,iter,resvec] = symmlq(A,b,···)：resvec 表示每次迭代的残差：norm(b−A*x0)。
- [x,flag,relres,iter,resvec,resveccg] = symmlq(A,b,···)：resveccg 表示每次迭代共轭梯度残差的范数。

【功能介绍】 求解特殊线性方程组。

【实例 2.104】 求解方程组 $\begin{cases} 2x_1+3x_2+1x_3=4 \\ 3x_1+4x_2+5x_3=7 \\ 1x_1+5x_2+6x_3=9 \end{cases}$。

```
>> A=[2,3,1;3,4,5;1,5,6]
A =
     2          3          1
     3          4          5
     1          5          6
>> b=[4,7,9]'
b =
     4
     7
     9
>> x = symmlq(A,b)
symmlq converged at iteration 2 to a solution with
relative residual 7.4e-017
x =
```

```
   -2/15
   19/15
    7/15
```

【实例讲解】 如果用户希望了解关于线性方程组中系数矩阵的性质，可以使用 symmlq 中比较复杂的函数形式。

2.4.7 bicg 函数——双共轭梯度法解方程组

【语法说明】

- x = bicg(A,b)：求线性方程组 AX=*b* 的解 X。A 必须为 n 阶方阵，b 为 n 元列向量。A 可以是由 afun 定义并返回 A*X 的函数。如果收敛，将显示结果信息；如果收敛失败，将给出警告信息并显示相对残差 norm(b−A*x)/norm(b)和计算终止的迭代次数。
- bicg(A,b,tol)：指定误差 tol，默认值是 1e−6。
- bicg(A,b,tol,maxit) ：maxit 指定最大迭代次数
- bicg(A,b,tol,maxit,M)：M 为用于对称正定矩阵的预处理因子。
- bicg(A,b,tol,maxit,M1,M2)：M=M1×M2。
- bicg(A,b,tol,maxit,M1,M2,x0)：x0 为初始估计值，默认值为 0。
- [x,flag] = bicg(A,b,…)：flag 的取值为 0 表示在指定迭代次数之内按要求精度收敛，为 1 表示在指定迭代次数内不收敛，为 2 表示 M 为坏条件的预处理因子，为 3 表示两次连续迭代完全相同，为 4 表示标量参数太小或太大。
- [x,flag,relres] = bicg(A,b,…)：relres 表示相对误差 norm(b−A*x)/norm(b)。
- [x,flag,relres,iter] = bicg(A,b,…) ：iter 表示计算 x 的迭代次数。
- [x,flag,relres,iter,resvec] = bicg(A,b,…)：resvec 表示每次迭代的残差：norm(b−A*x0)。

【功能介绍】 用 bicg 函数求解方程组的解。

【实例 2.105】 求解方程组 $\begin{cases} 2x_1+3x_2+1x_3=4 \\ 3x_1+4x_2+5x_3=7 \\ 1x_1+5x_2+6x_3=9 \end{cases}$ 的解。

```
>> A=[2,3,1;3,4,5;1,5,6]

A =

     2     3     1
     3     4     5
     1     5     6

>>  b=[4,7,9]'

b =

     4
     7
     9

>> x = bicg(A,b)
bicg converged at iteration 3 to a solution with relative
residual 4e-016

x =

   -0.1333
    1.2667
    0.4667

>> tol=0.001

tol =

  1.0000e-003

>> maxit=10000

maxit =

       10000

>> bicg(A,b,tol,maxit)          %指定了误差和迭代次数
bicg converged at iteration 3 to a solution with relative
```

```
residual 4e-016
    ans =

       -0.1333
        1.2667
        0.4667
```

【实例讲解】 指定迭代误差和迭代次数的计算结果与没有指定的计算结果是相同的，说明在还没有达到迭代次数时已经收敛。

2.4.8 bicgstab 函数——稳定双共轭梯度方法解方程组

【语法说明】 稳定双共轭梯度法解方程组，调用方式和返回的结果形式与函数 bicg 一样。

- x =bicgstab(A,b)。
- bicgstab(A,b,tol)。
- bicgstab(A,b,tol,maxit)。
- bicgstab(A,b,tol,maxit,M)。
- bicgstab(A,b,tol,maxit,M1,M2)。
- bicgstab(A,b,tol,maxit,M1,M2,x0)。
- [x,flag] = bicgstab(A,b,…)。
- [x,flag,relres] = bicgstab(A,b,…)。
- [x,flag,relres,iter] = bicgstab(A,b,…)。
- [x,flag,relres,iter,resvec] = bicgstab(A,b,…)。

【功能介绍】 用 bicgstab 函数求解方程组的解。

【实例 2.106】 求方程组 $\begin{cases} 2x_1+3x_2+1x_3=4 \\ 3x_1+4x_2+5x_3=7 \\ 1x_1+5x_2+6x_3=9 \end{cases}$ 的解。

```
    >>  A=[2,3,1;3,4,5;1,5,6]
    A =

         2     3     1
         3     4     5
```

```
     1     5     6

>>  b=[4,7,9]'

b =

     4
     7
     9

>> x =bicgstab(A,b)
bicgstab converged at iteration 2.5 to a solution with
relative residual 7.4e-017

x =

   -0.1333
    1.2667
    0.4667

>> bicgstab(A,b,0.001,20000)
bicgstab converged at iteration 2.5 to a solution with
relative residual 7.4e-017

ans =

   -0.1333
    1.2667
    0.4667
```

【实例讲解】 稳定双共轭梯度法是求解线性方程组的一个重要方法，关于这个方法的具体理论和使用方法，用户可以参考相关的矩阵书籍。

2.4.9　cgs 函数——复共轭梯度平方法解方程组

【语法说明】 cgs 函数调用方式、返回结果的形式与函数 bicg 一样。

- x = cgs(A,b)。

- cgs(A,b,tol)。
- cgs(A,b,tol,maxit)。
- cgs(A,b,tol,maxit,M)。
- cgs(A,b,tol,maxit,M1,M2)。
- cgs(A,b,tol,maxit,M1,M2,x0)。
- [x,flag] = cgs(A,b,…)。
- [x,flag,relres] = cgs(A,b,…)。
- [x,flag,relres,iter] = cgs(A,b,…)。
- [x,flag,relres,iter,resvec] = cgs(A,b,…)。

【功能介绍】 用复共轭梯度平方法解方程组。

【实例 2.107】 求方程组 $\begin{cases} 2x_1+3x_2+1x_3=4 \\ 3x_1+4x_2+5x_3=7 \\ 1x_1+5x_2+6x_3=9 \end{cases}$ 的解。

```
>> A=[2,3,1;3,4,5;1,5,6]
A =

     2     3     1
     3     4     5
     1     5     6

>> b=[4,7,9]'

b =

     4
     7
     9
>> x = cgs(A,b)
cgs converged at iteration 3 to a solution with relative
residual 9.1e-014

x =

   -0.1333
    1.2667
    0.4667
```

【实例讲解】 通过上面几个例子可以看出，无论用哪种方法计算，求得的方程组的解都是相同的。

2.4.10 lsqr 函数——共轭梯度的 LSQR 方法

【语法说明】 调用方式和返回结果的形式与函数 bicg 一样。

- x = lsqr(A,b)。
- lsqr(A,b,tol)。
- lsqr(A,b,tol,maxit)。
- lsqr(A,b,tol,maxit,M)。
- lsqr(A,b,tol,maxit,M1,M2)。
- lsqr(A,b,tol,maxit,M1,M2,x0)。
- [x,flag] = lsqr(A,b,…)。
- [x,flag,relres] = lsqr(A,b,…)。
- [x,flag,relres,iter] = lsqr(A,b,…)。
- [x,flag,relres,iter,resvec] = lsqr(A,b,…)。

【功能介绍】 LSQR 方法解方程组。

【实例 2.108】 用 LSQR 判断解方程组，判断收敛性。

```
>> N1 = ones(60,1);
>> A = spdiags([-2*N1 4*N1 -N1],-1:1,60,60);    %产生一个对角矩阵
>> b = sum(A,2);
>> tol = 1e-8;     %指定精度
>> maxit = 15;     %指定最大迭代次数
>> M1 = spdiags([N1/(-2) N1],-1:0,60,60);
>> M2 = spdiags([4*N1 -N1],0:1,60,60);
>> [x,flag,relres,iter,resvec] = lsqr(A,b,tol, maxit, M1,M2,[])
```

上面函数得到的结果是：

```
x = (全是 1)
flag =
    0
relres =
```

```
  3.6057e-009
iter =
    11
resvec =
    8.4261
    2.3361
    0.3746
    0.0615
    0.0103
    0.0017
    0.0003
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
```

【实例讲解】 读者可以修改精度和迭代次数数值，查看 LSQR 函数计算的结果。

2.4.11 qmres 函数——广义最小残差法

【语法说明】 参数 restart（默认时为系数方阵 *A* 的阶数 *n*）可以给出；调用方式和返回结果的形式与函数 bicg 一样。

- x = gmres(A,b)。
- gmres(A,b,restart)。
- gmres(A,b,restart,tol)。
- gmres(A,b,restart,tol,maxit)。
- gmres(A,b,restart,tol,maxit,M)。
- gmres(A,b,restart,tol,maxit,M1,M2)。
- gmres(A,b,restart,tol,maxit,M1,M2,x0)。
- [x,flag] = gmres(A,b,…)。
- [x,flag,relres] = gmres(A,b,…) 。
- [x,flag,relres,iter] = gmres(A,b,…)。
- [x,flag,relres,iter,resvec] = gmres(A,b,…)。

【功能介绍】 用广义最小残差法解方程组。

【实例 2.109】 求方程组 $\begin{cases} 6x_1+3x_2+1x_3=9 \\ 3x_1-4x_2+5x_3=-7 \\ 2.9x_1+5x_2+6x_3=14.6 \end{cases}$ 的解。

```
>> A=[6 3 1;3 -4 5;2.9 5 6]

A =

    6.0000    3.0000    1.0000
    3.0000   -4.0000    5.0000
    2.9000    5.0000    6.0000

>> b=[9 -7 14.6]'

b =

    9.0000
   -7.0000
   14.6000

>> x = gmres(A,b)
gmres converged at iteration 3 to a solution with
relative residual 2.9e-016

x =

    0.2612
    2.3656
    0.3357

>> gmres(A,b,3,0.001,20000)  %阶数指定为 3
??? Attempted to access w(4); index out of bounds because
numel(w)=3.

Error in ==> gmres at 328
        normr = abs(w(initer+1));

>> gmres(A,b,2,0.001,20000)
```

```
gmres(2) converged at outer iteration 3 (inner iteration
2) to a solution with relative residual 0.00063
ans =

    0.2590
    2.3660
    0.3370
```

【实例讲解】 在这个例子中，当阶数为 3 时报错，即警告指定的阶数不能超过系数矩阵的阶数。

2.4.12 minres 函数——最小残差法解方程组

【语法说明】 这里的矩阵 ***A*** 为对称矩阵，最小残差法解方程组的方法是通过寻找最小残差来求 *x*。调用方式和返回的结果形式与函数 bicg 一样。

- x = minres(A,b)。
- minres(A,b,tol)。
- minres(A,b,tol,maxit)。
- minres(A,b,tol.maxit,M)。
- minres(A,b,tol,maxit,M1,M2)。
- minres(A,b,tol,maxit,M1,M2,x0)。
- [x,flag] = minres(A,b,⋯)。
- [x,flag,relres] = minres(A,b,⋯)。
- [x,flag,relres,iter] = minres(A,b,⋯)。
- [x,flag,relres,iter,resvec] = minres(A,b,⋯)。
- [x,flag,relres,iter,resvec,resveccg] = minres(A,b,⋯)。

【功能介绍】 用最小残差法解方程组。

【实例 2.110】 用最小残差法求解方程组。

```
>> N1 = ones(60,1);
>> A = spdiags([-2*N1 4*N1 -2*N1],-1:1,60,60);
>> b = sum(A,2);
>> tol = 1e-6;
>> maxit = 30;
```

```
>> M1 = spdiags(4*N1,0,60,60);
>> [x,flag,relres,iter,resvec,resveccg] = minres (A,b,tol,
maxit,M1,[],[])
```

上面的函数得到的结果为：

```
x = (全是 1)
   flag =
    0
relres =
  1.6921e-014
iter =
   29
resvec =
    2.8284
    1.2649
    0.7559
    0.5164
    0.3814

resveccg =
    2.8284
    1.4142
    0.9428
    0.7071
    0.5657
```

【实例讲解】 在这个例子中，首先使用 spdiags 函数创建一个稀疏对角阵，然后再使用 mineres 函数求解方程组的结果。

2.4.13 pcg 函数——预处理共轭梯度方法

【语法说明】 这里 *A* 为对称正定矩阵，调用方式和返回的结果形式与函数 bicg 一样。

- x = pcg(A,b)。
- pcg(A,b,tol)。
- pcg(A,b,tol,maxit)。
- pcg(A,b,tol,maxit,M)。
- pcg(A,b,tol,maxit,M1,M2)。

- pcg(A,b,tol,maxit,M1,M2,x0)。
- pcg(A,b,tol,maxit,M1,M2,x0,p1,p2,…)。
- [x,flag] = pcg(A,b,tol,maxit,M1,M2,x0,p1,p2,…) 。
- [x,flag,relres] = pcg(A,b,tol,maxit,M1,M2,x0,p1,p2,…)。
- [x,flag,relres,iter] = pcg(A,b,tol,maxit,M1,M2,x0,p1,p2,…)。
- [x,flag,relres,iter,resvec] = pcg(A,b,tol,maxit,M1,M2,x0,p1,p2,…)。

【功能介绍】 用预处理共轭梯度法解方程组。

【实例 2.111】 求方程组 $\begin{cases} 2x_1 + 1x_2 + 1x_3 = 3 \\ 1x_1 + 2x_2 + 1x_3 = 4 \\ 1x_1 + 1x_2 + 2x_3 = 5 \end{cases}$ 的解。

```
>> A=[2 1 1;1 2 1;1 1 2]

A =

     2     1     1
     1     2     1
     1     1     2

>> b=[3 4 5]'

b =

     3
     4
     5

>> x = pcg(A,b)
pcg converged at iteration 2 to a solution with relative
residual 3.8e-016

x =

    0.0000
    1.0000
    2.0000
```

【实例讲解】 这个函数要求系数矩阵一定是正定矩阵，否则系统会发出警告错误。

2.4.14 qmr 函数——准最小残差法解方程组

【语法说明】 这里 ***A*** 为方阵，调用方式和返回的结果形式与函数 bicg 一样。

- x = qmr(A,b)。
- qmr(A,b,tol)。
- qmr(A,b,tol,maxit)。
- qmr(A,b,tol,maxit,M)。
- qmr(A,b,tol,maxit,M1,M2)。
- qmr(A,b,tol,maxit,M1,M2,x0)。
- qmr(afun,b,tol,maxit,m1fun,m2fun,x0,p1,p2,⋯)。
- [x,flag] = qmr(A,b,⋯)。
- [x,flag,relres] = qmr(A,b,⋯)。
- [x,flag,relres,iter] = qmr(A,b,⋯)。
- [x,flag,relres,iter,resvec] = qmr(A,b,⋯)。

【功能介绍】 用准最小残差法解方程组。

【实例 2.112】 求方程组 $\begin{cases} 2x_1 + 1x_2 + 1x_3 = 3 \\ 1x_1 + 2x_2 + 1x_3 = 4 \\ 1x_1 + 1x_2 + 2x_3 = 5 \end{cases}$ 的解。

```
>> A=[2 1 1;1 2 1;1 1 2]

A =

     2     1     1
     1     2     1
     1     1     2

>> b=[3 4 5]'

b =

     3
     4
```

```
    5
>> x = qmr(A,b)
qmr converged at iteration 2 to a solution with relative
residual 1.8e-016

x =

    0.0000
    1.0000
    2.0000
```

【实例讲解】 根据前面的定义，使用这个函数求解方程组解的时候，系数矩阵必须是方阵。

2.5 特征值与二次型

在许多数值计算和工程技术问题中，特征值和特征向量都占有极为重要的地位，如工程中的振动问题和稳定性问题，常归结为求一个方阵的特征值和特征向量。本节将要讲解与特征值相关的函数。

2.5.1 特征值与特征向量的求法

特征值与特征向量的定义：设 $\boldsymbol{A}$ 为 n 阶方阵，如果有数值“λ”与 n 维列向量 $\boldsymbol{x}$ 使得关系式 $\boldsymbol{Ax}=\lambda\boldsymbol{x}$ 成立，则称 λ 为方阵 $\boldsymbol{A}$ 的特征值，非零向量 $\boldsymbol{x}$ 称为 $\boldsymbol{A}$ 对应于特征值“λ”的特征向量。

【实例 2.113】 求矩阵 $\boldsymbol{A}=\begin{pmatrix}-3 & 4 & 0\\ -4 & 3 & 7\\ 2 & -5 & 9\end{pmatrix}$ 的特征值和特征向量。

```
>> A=[-3 4 0;-4 3 7;2 -5 9]
A =
    -3     4     0
    -4     3     7
     2    -5     9
>> [V,D]=eig(A)
```

```
V =
   0.7516              0.3095 - 0.1902i   0.3095 + 0.1902i
   0.6313              0.7803              0.7803
   0.1914              0.3240 + 0.3927i   0.3240 - 0.3927i
D =
   0.3600                  0                  0
        0            4.3200 + 4.4975i         0
        0                  0            4.3200 - 4.4975i
```

【实例讲解】 这个例子中特征值和特征向量既有实数又有复数，并且复数特征值为一对共轭的复数。

【实例 2.114】 求矩阵 $A=\begin{pmatrix} 3 & -2 & 1 \\ 0 & 2 & 8 \\ -5 & 1 & 0 \end{pmatrix}$ 的特征值和特征向量。

```
>> A=[-3  -2  1;0  2  8;-5  1  0]
A =
    -3    -2     1
     0     2     8
    -5     1     0
>> [V,D]=eig(A)
V =
   0.0932 + 0.4973i   0.0932 - 0.4973i  -0.1818
   0.6815              0.6815              0.9163
  -0.4309 + 0.3064i  -0.4309 - 0.3064i   0.3568
D =

  -3.0578 + 3.5965i          0                  0
        0            -3.0578 - 3.5965i          0
        0                    0              5.1156
```

【实例讲解】 在这个例子中得到的特征值与特征向量包含复数的元素。

2.5.2 cdf2rdf 函数——复对角矩阵转化为实对角矩阵

【语法说明】 [V,D] = cdf2rdf (v,d)：将复对角阵 v、d 变为实对角阵 V、D，在对角线上，用 2×2 实数块代替共轭复数对。

【功能介绍】 将复对角矩阵转化为实对角矩阵。

【实例 2.115】 先对矩阵 $A=\begin{bmatrix}1 & 2 & 3\\0 & 4 & 5\\0 & -5 & 4\end{bmatrix}$ 求特征值和特征向量，然后对复对角矩阵进行转化。

```
>> A=[1 2 3;0 4 5;0 -5 4];
>> [v,d]=eig(A)
v =
    1.0000              -0.0191 - 0.4002i  -0.0191 + 0.4002i
         0                   0 - 0.6479i        0 + 0.6479i
         0               0.6479              0.6479
d =
    1.0000                   0                   0
         0          4.0000 + 5.0000i             0
         0                   0          4.0000 - 5.0000i
>> [V,D]=cdf2rdf(v,d)
V =
    1.0000   -0.0191   -0.4002
         0         0   -0.6479
         0    0.6479         0
D =
    1.0000         0         0
         0    4.0000    5.0000
         0   -5.0000    4.0000
```

【实例讲解】 关于共轭复数对的概念，请读者查看相关的书籍。

2.5.3 orth 函数——将矩阵正交规范化

【语法说明】 B=orth(A)：将矩阵 A 正交规范化，B 的列与 A 的列具有相同的空间，B 的列向量是正交向量，且满足：B'*B = eye(rank(A))。

【功能介绍】 把矩阵正交规范化。

【实例 2.116】 将矩阵 $A=\begin{pmatrix}4 & 0 & 0\\0 & 3 & 1\\0 & 1 & 3\end{pmatrix}$ 正交规范化。

```
>>A=[4  0  0; 0  3  1; 0  1  3];
A =
```

```
    4     0     0
    0     3     1
    0     1     3
>>B=orth(A)
B =
          0    1.0000         0
   -0.7071         0   -0.7071
   -0.7071         0    0.7071
>>Q=B'*B
```

上面函数的结果为：

```
P =
    1.0000         0         0
         0    0.7071   -0.7071
         0    0.7071    0.7071
Q =
    1.0000         0         0
         0    1.0000         0
         0         0    1.0000
```

【实例讲解】 从上面的结果中可以看出，矩阵 ***B*** 和 ***B*** 转置的乘积是单位矩阵。

2.6 秩与线性相关性

在线性代数中，相关性使用得非常广泛，而且，通常用向量组的秩来判断线性相关性。在下面的小节中，将简单介绍 MATLAB 函数中怎样利用秩判断线性相关。

2.6.1 利用 rank 函数判断矩阵和向量组的秩以及向量组的线性相关性

【语法说明】

- k = rank(A)：返回矩阵 A 的行（或列）向量中线性无关个数。
- k = rank(A,tol) ：与上相同，tol 为给定误差。

【功能介绍】 矩阵 A 的秩是矩阵 A 中最高阶非零子式的阶数；

向量组的秩通常由该向量组构成的矩阵来计算。

【实例2.117】 求向量组$\alpha_1=(1\quad -2\quad 2\quad 3)$，$\alpha_2=(-2\quad 4\quad -1\quad 3)$，$\alpha_3=(-1\quad 2\quad 0\quad 3)$，$\alpha_4=(0\quad 6\quad 2\quad 3)$，$\alpha_5=(2\quad -6\quad 3\quad 4)$的秩，并判断其线性相关性。

```
>>A=[1 -2 2 3;-2 4 -1 3;-1 2 0 3;0 6 2 3;2 -6 3 4];
A =
     1    -2     2     3
    -2     4    -1     3
    -1     2     0     3
     0     6     2     3
     2    -6     3     4
>>k=rank(A)
```

上面函数得到的结果为：

```
k =
     3
```

【实例讲解】 在这个例子中，由于秩为 3 < 向量个数，因此向量组线性相关。

2.6.2 求行阶梯矩阵及向量组的基

【语法说明】

- R = rref(A)：用高斯-约当消元法和行主元法求 A 的行最简行矩阵 R。
- [R,jb] = rref(A)：jb 是一个向量，r = length(jb)为 A 的秩，A(:, jb)为 A 的列向量基；jb 中元素表示基向量所在的列。
- [R,jb] = rref(A,tol)：tol 为指定的精度。
- rrefmovie(A)：给出每一步化简的过程。

【功能介绍】 用 rref 函数求最大无关组和向量基。

【实例 2.118】 求向量组 *a*1=(1,-3,4,2)，*a*2=(-2,5,-3,3)，*a*3=(3,2,5,0)，*a*4=(0,5,3,4)，*a*5=(2,-6,3,4)的一个最大无关组。

```
>> a1=[1,-3,4,2]
a1 =
     1    -3     4     2
>> a2=[-2,5,-3,3];
```

```
>> a3=[3,2,5,0];
>> a4=[0,5,3,4];
>> a5=[2,-6,3,4];
>> A=[a1  a2  a3  a4  a5]
A =
  Columns 1 through 14
     1    -3     4     2    -2     5    -3     3     3     2
5     0     0     5
  Columns 15 through 20
     3     4     2    -6     3     4
>>  [R,jb]=rref(A)
R =
  Columns 1 through 14
     1    -3     4     2    -2     5    -3     3     3     2
5     0     0     5
  Columns 15 through 20
     3     4     2    -6     3     4
jb =
     1
>> B = A(:,jb)
B =
     1
```

【实例讲解】 这个例子中 $a1$ 为向量组的一个基。

2.7　稀疏矩阵技术

稀疏矩阵是比较特殊的矩阵形式，但是在工程或理论运算中应用得极为广泛，在下面的小节中，详细讲解有关系数矩阵技术的内容。包括稀疏矩阵的转化，稀疏矩阵内部元素之间的变换、排序等。

2.7.1　sparse 函数——创建稀疏矩阵

【语法说明】

- S = sparse(A)：将矩阵 A 转化为稀疏矩阵形式，即由 A 的非零元素和下标构成稀疏矩阵 S。若 A 本身为稀疏矩阵，则返回 A

本身。

- S = sparse(m,n)：生成一个 m×n 的所有元素都是 0 的稀疏矩阵。
- S = sparse(i,j,s)：生成一个由长度相同的向量 i、j 和 s 定义的稀疏矩阵 S，其中 i，j 是整数向量，定义稀疏矩阵的元素位置(i,j)，s 是一个标量或与 i、j 长度相同的向量，表示在(i,j)位置上的元素。
- S = sparse(i,j,s,m,n)：生成一个 m×n 的稀疏矩阵，(i,j)对应位置元素为 s_i，m = max(i)且 n =max(j)。
- S = sparse(i,j,s,m,n,nzmax)：生成一个 m×n 的含有 nzmax 个非零元素的稀疏矩阵 S，nzmax 的值必须大于或者等于向量 i 和 j 的长度。

【功能介绍】 用 sparse 函数生成稀疏矩阵。

【实例 2.119】 生成稀疏矩阵。

```
>> S=sparse(1:10,1:10,1:10)
S =
   (1,1)        1
   (2,2)        2
   (3,3)        3
   (4,4)        4
   (5,5)        5
   (6,6)        6
   (7,7)        7
   (8,8)        8
   (9,9)        9
  (10,10)      10
>> S=sparse(1:10,1:10,5)
S =
   (1,1)        5
   (2,2)        5
   (3,3)        5
   (4,4)        5
   (5,5)        5
   (6,6)        5
   (7,7)        5
   (8,8)        5
```

```
   (9,9)         5
  (10,10)        5
```

【实例讲解】 从上面的结果中可以看出，MATLAB 给出的是稀疏矩阵中非零元素的位置和数值。

2.7.2 full 函数——将稀疏矩阵转化为满矩阵

【语法说明】 A=full(S) ：S 为稀疏矩阵，A 为满矩阵。

【功能介绍】 稀疏矩阵转化为满矩阵。

【实例 2.120】 将稀疏矩阵转化为满矩阵。

```
>> S=sparse(1:5,1:5,4:8)
S =
   (1,1)        4
   (2,2)        5
   (3,3)        6
   (4,4)        7
   (5,5)        8
>> A=full(S)
A =
     4     0     0     0     0
     0     5     0     0     0
     0     0     6     0     0
     0     0     0     7     0
     0     0     0     0     8
```

【实例讲解】 上面例子中的矩阵 A 就是正常的满矩阵，而不是稀疏矩阵。

2.7.3 find 函数——稀疏矩阵非零元素的索引

【语法说明】

- k = find(x) ：按行检索 X 中非零元素的点，若没有非零元素，将返回空矩阵。
- [i,j] = find(X)：检索 X 中非零元素的行标 i 和列标 j。
- [i,j,v] = find(X)：检索 X 中非零元素的行标 i 和列标 j 以及对应的元素值 v。

【实例 2.121】 求解稀疏矩阵中非零元素的索引。

```
>> S=sparse(1:6,1:6,4:9)
S =
   (1,1)        4
   (2,2)        5
   (3,3)        6
   (4,4)        7
   (5,5)        8
   (6,6)        9
>>  [i,j,v]=find(S)
```

上面函数得到的结果为：

```
i =
     1
     2
     3
     4
     5
     6
j =
     1
     2
     3
     4
     5
     6
v =
     4
     5
     6
     7
     8
     9
```

【实例讲解】 通过使用 find 函数，用户可以很便利地确定稀疏矩阵中非零元素的信息。

2.7.4 spconvert 函数——外部数据转化为稀疏矩阵

【语法说明】 S=spconvert(D) ：D 是只有 3 列或 4 列的矩阵。

【功能介绍】 将外部数据转化为稀疏矩阵。

【实例 2.122】 将下面的两个外部数据转化为稀疏矩阵。

```
>> A=[1  2  3;2  5  4;3  4  6;3  6  7]
A =
     1     2     3
     2     5     4
     3     4     6
     3     6     7
>> S=spconvert(A)
S =
   (1,2)        3
   (3,4)        6
   (2,5)        4
   (3,6)        7
>> B=[1 2 3 4; 2 5 4 0;3 4 6 9;3 6 7 4];
B =
     1     2     3     4
     2     5     4     0
     3     4     6     9
     3     6     7     4
>> S=spconvert(A)
S =
   (1,2)      3.0000 + 4.0000i
   (3,4)      6.0000 + 9.0000i
   (2,5)      4.0000
   (3,6)      7.0000 + 4.0000i
```

【实例讲解】 待转化的矩阵必须满足 3 行或者 4 列的条件，***A*** 矩阵是 3×4 的矩阵，满足这个条件，***B*** 矩阵是 4×4 的矩阵，满足 4 列的条件。

2.7.5 spdiags 函数——生成带状（对角）稀疏矩阵

【语法说明】

- [B,d] = spdiags(A)：从矩阵 A 中取出所有非零对角元素，并保存在矩阵 B 中，向量 d 表示非零元素的对角线位置。
- B = spdiags(A,d)：从 A 中取出由 d 指定的对角线元素，并保存在 B 中。
- A = spdiags(B,d,A)：用 B 中的列替换 A 中由 d 指定的对角线元素，输出稀疏矩阵。

- A = spdiags(B,d,m,n)：产生一个 m×n 稀疏矩阵 A，其元素是 B 中的列元素放在由 d 指定的对角线位置上。

【功能介绍】 建立带状稀疏矩阵。

【实例 2.123】 建立一个稀疏矩阵，并指出取出的元素在原矩阵中的位置。

```
>>A =[11   0   13   0
      0   22   0   24
      0    0  33    0
      41   0    0  44
      0   52    0   0
      0    0  63    0
      0    0    0  74];
>>[B,d] = spdiags(A)
B =
    41    11     0
    52    22     0
    63    33    13
    74    44    24
d =
    -3       %B 的第 1 列元素在 A 中主对角线下方第 3 条对角线上
     0       %B 的第 2 列在 A 的主对角线上
     2       %B 的第 3 列在 A 的主对角线上方第 2 条对角线上
```

【实例讲解】 从上面的结果可以看出，数值 d 通过一个数值来表示数值的位置。

2.7.6 speye 函数——单位稀疏矩阵

【语法说明】

- S = speye(m,n)：生成 m×n 的单位稀疏矩阵。
- S = speye(n)：生成 n×n 的单位稀疏矩阵。

【功能介绍】 建立单位稀疏矩阵。

【实例 2.124】 用 speye 函数生成稀疏矩阵。

```
>> S=speye(4,5)
S =
   (1,1)        1
   (2,2)        1
   (3,3)        1
```

```
   (4,4)        1
>> S1=(5)
S1 =
    5
```

【实例讲解】 单位稀疏矩阵和单位矩阵类似，只是维度比较大。

2.7.7　sprand 函数——稀疏均匀分布随机矩阵

【语法说明】

- R = sprand(S)：生成与 S 具有相同稀疏结构的均匀分布随机矩阵。
- R = sprand(m,n,density) ：生成一个 m×n 的服从均匀分布的随机稀疏矩阵，非零元素的分布密度是 density。
- R = sprand(m,n,density,rc) ：生成一个近似的条件数为 1/rc、大小为 m×n 的均匀分布的随机稀疏矩阵。

【功能介绍】 稀疏均匀分布随机矩阵。

【实例 2.125】 创建不同维度的随机稀疏矩阵。

```
>> S=speye(4,5)
S =
   (1,1)        1
   (2,2)        1
   (3,3)        1
   (4,4)        1
>> R = sprand(S)
R =
   (1,1)      0.9501
   (2,2)      0.2311
   (3,3)      0.6068
   (4,4)      0.4860
```

【实例讲解】 用户可以将上面的计算结果和 rand 函数产生的结果进行对比分析。

2.7.8　sprandn 函数——生成稀疏正态分布随机矩阵

【语法说明】

- R = sprandn(S)：生成与 S 具有相同稀疏结构的正态分布随

机矩阵。

- R = sprandn(m,n,density)：生成一个 m×n 的服从正态分布的随机稀疏矩阵，非零元素的分布密度是 density。
- R = sprandn(m,n,density,rc) ：生成一个近似的条件数为 1/rc、大小为 m×n 的均匀分布的随机稀疏矩阵。

【功能介绍】 建立稀疏正态分布随机矩阵。

【实例 2.126】 创建正态分布随机矩阵。

```
>>  S=speye(4,5)
S =
   (1,1)       1
   (2,2)       1
   (3,3)       1
   (4,4)       1
>> sprandn(S)
ans =
   (1,1)      -0.4326
   (2,2)      -1.6656
   (3,3)       0.1253
   (4,4)       0.2877
```

【实例讲解】 关于多维正态分布的概念，用户可以查看相应的书籍。

2.7.9 sprandsym 函数——稀疏对称随机矩阵

【语法说明】

- R = sprandsym(S)：生成稀疏对称随机矩阵，其下三角和对角线与 S 具有相同的结构，其元素服从均值为 0、方差为 1 的标准正态分布。
- R = sprandsym(n,density)：生成 n×n 的稀疏对称随机矩阵，矩阵元素服从正态分布，分布密度为 density。
- R = sprandsym(n,density,rc)：生成近似条件数为 1/rc 的稀疏对称随机矩阵。
- R = sprandsym(n,density,rc,kind)：生成一个正定矩阵，参数 kind 取值为 kind=1 表示矩阵由一正定对角矩阵经随机雅克比转换得

到，其条件数正好为 1/rc；kind=2 表示矩阵为外积的换位和，其条件数近似等于 1/rc；kind=3 表示生成一个与矩阵 S 结构相同的稀疏随机矩阵，条件数近似为 1/rc。

【实例 2.127】 产生均值为 0、方差为 1 的对称随机矩阵和 6×6 的稀疏对称随机矩阵，矩阵元素服从正态分布，分布密度为 0.3。

```
>> S=speye(6,6)

S =

   (1,1)        1
   (2,2)        1
   (3,3)        1
   (4,4)        1
   (5,5)        1
   (6,6)        1

>> R = sprandsym(S)

R =

   (1,1)       -0.4326
   (2,2)       -1.6656
   (3,3)        0.1253
   (4,4)        0.2877
   (5,5)       -1.1465
   (6,6)        1.1909

>> R = sprandsym(6,0.3)

R =

   (2,1)        0.1746
   (1,2)        0.1746
   (3,3)        1.1892
   (6,3)        0.3273
   (5,4)       -0.1867
   (6,4)        0.7258
   (4,5)       -0.1867
   (5,5)       -0.0376
```

```
    (3,6)        0.3273
    (4,6)        0.7258
```

【实例讲解】 关于多维正态分布的概念，用户可以查看相应的书籍。

2.7.10 nnz 函数——返回稀疏矩阵非零元素的个数

【语法说明】 n = nnz(X)：返回矩阵 X 中非零元素的个数。

【功能介绍】 计算稀疏矩阵非零元素的个数。

【实例 2.128】 统计稀疏矩阵的非零元素的个数。

```
>> S=speye(5,6)
S =
    (1,1)        1
    (2,2)        1
    (3,3)        1
    (4,4)        1
    (5,5)        1
>> nnz(S)
ans =
     5
```

【实例讲解】 对于稀疏矩阵而言，非零元素的个数是一个重要信息。

2.7.11 nonzeros 函数——找到稀疏矩阵的非零元素

【语法说明】 s = nonzeros(A)：返回矩阵 A 中非零元素按列顺序构成的列向量。

【功能介绍】 列出稀疏矩阵中的非零元素。

【实例 2.129】 找到矩阵 $\boldsymbol{A}=\begin{bmatrix}1 & 3 & 0 & 8\\0 & 2 & 6 & 9\\2 & 0 & 0 & 6\\0 & 5 & 9 & 2\end{bmatrix}$ 中的非零元素。

```
>> A=[1,3,0,8;0,2,6,9;2,0,0,6;0,5,9,2]
A =
     1     3     0     8
```

```
     0     2     6     9
     2     0     0     6
     0     5     9     2
>> s=nonzeros(A)
s =
     1
     2
     3
     2
     5
     6
     9
     8
     9
     6
     2
```

【实例讲解】 这个函数可以和前面函数结合起来使用，确定非零元素的位置和具体数值。

2.7.12 nzmax 函数——稀疏矩阵非零元素的内存分配

【语法说明】 n = nzmax(S)：返回非零元素分配的内存总数 n。

【功能介绍】 计算非零元素分配的内存总数。

【实例 2.130】 计算矩阵 $\boldsymbol{A}=\begin{bmatrix}1 & 3 & 0 & 8\\0 & 2 & 6 & 9\\2 & 0 & 0 & 6\\0 & 5 & 9 & 2\end{bmatrix}$ 中非零元素分配的内存总数。

```
>> A=[1,3,0,8;0,2,6,9;2,0,0,6;0,5,9,2]
A =
     1     3     0     8
     0     2     6     9
     2     0     0     6
     0     5     9     2
>> n = nzmax(A)
n =
    16
```

【实例讲解】 内存总数的分配，对于稀疏矩阵而言，是一个重要的信息，会直接影响运算的效率。

2.7.13 spfun 函数——稀疏矩阵的非零元素应用

【语法说明】 f = spfun('function',S)：用 S 中非零元素对函数 function 求值，如果 function 不是对稀疏矩阵定义的，同样可以求值。

【实例 2.131】 指数 e 的方幂求解。

```
%4 阶稀疏矩阵对角矩阵
S =
   (1,1)        1
   (2,2)        2
   (3,3)        3
   (4,4)        4
f= spfun('exp',S)     %即指数 e 的非零元素方幂
```

上面函数得到的结果为：

```
f =
   (1,1)       2.7183
   (2,2)       7.3891
   (3,3)      20.0855
   (4,4)      54.5982
```

【实例讲解】 读者可以尝试对稀疏矩阵的元素进行其他运算。

2.7.14 spy 函数——画稀疏矩阵非零元素的分布图形

【语法说明】

- spy(S)：画出稀疏矩阵 S 中非零元素的分布图形。S 也可以是满矩阵。
- spy(S,markersize)：markersize 为整数，指定点阵大小。
- spy(S,'LineSpec')：LineSpec 指定绘图标记和颜色。
- spy(S,'LineSpec',markersize)：参数与上面相同。

【实例 2.132】 加载 MATLAB 自带数据文件，画出其非零元素分布图形。

```
>> load west0479
>> A=west0479;
>> spy(A,'ro',3)
```

该例子画出的矩阵非零元素分布如图 2.1 所示。

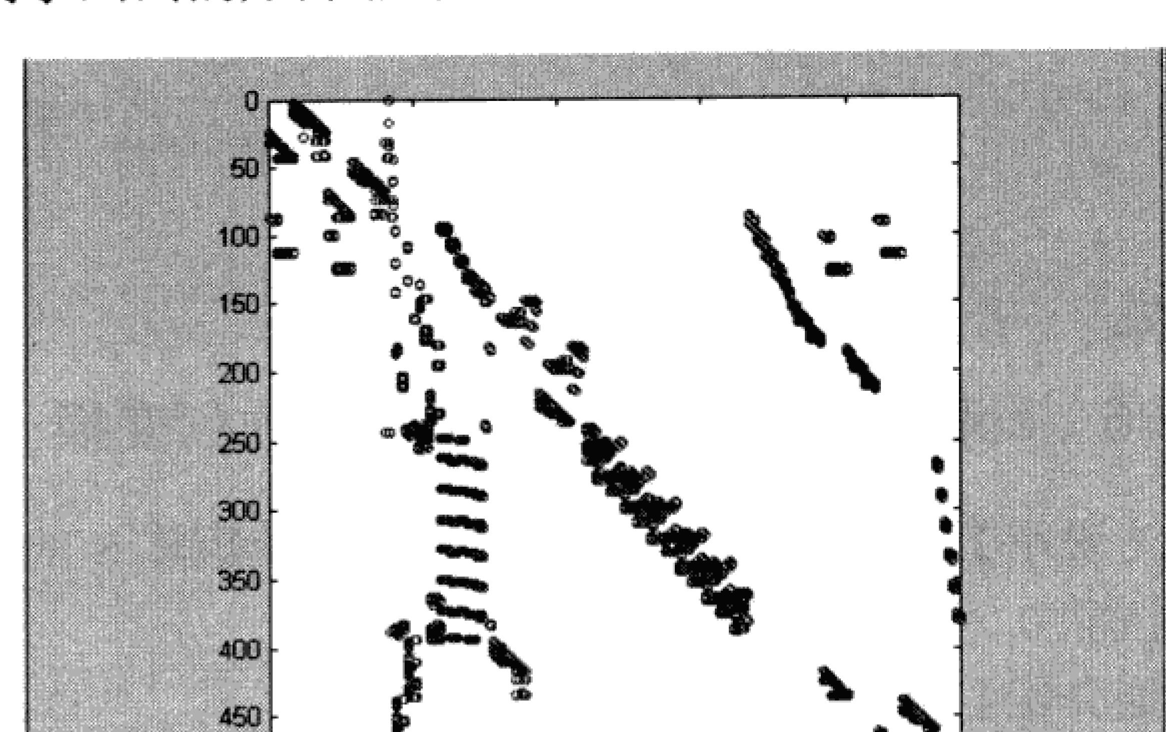


图 2.1 稀疏矩阵中非零元素的分布图

【实例讲解】 从上面的结果可以看出，非零元素的分布图可以直观地表现稀疏矩阵的元素分布。

2.7.15 colmmd 函数——稀疏矩阵的排序

【语法说明】 p = colmmd(S)：返回稀疏矩阵 S 的列的最小度排序向量 p，按 p 排列后的矩阵为 S(:, p)。

【功能介绍】 对稀疏矩阵排序。

【实例 2.133】 对稀疏矩阵进行排序。

```
>> S=speye(5,6)

S =

   (1,1)        1
   (2,2)        1
   (3,3)        1
   (4,4)        1
   (5,5)        1

>> p = colmmd(S)
```

```
    Warning: COLMMD is obsolete and will be removed in a
future version. Use COLAMD instead.
    > In colmmd at 15

    p =

         1     2     3     4     5     6
```

2.7.16 colperm 函数——非零元素的列变换

【语法说明】 j=colperm(S)：返回一个稀疏矩阵 S 的列变换的向量。列按非 0 元素升序排列。

【功能介绍】 对稀疏矩阵进行列变换。

【实例 2.134】 创建稀疏矩阵并返回其列变换向量。

```
>> S=speye(5,6)
S =
   (1,1)        1
   (2,2)        1
   (3,3)        1
   (4,4)        1
   (5,5)        1
>> j = colperm(S)
j =
     6     1     2     3     4     5
```

2.7.17 dmperm 函数——Dulmage-Mendelsohn 分解

【语法说明】

- p = dmperm (A)：返回 A 的行排列向量 p，这样，如果 A 满列秩，就使得 A(p,:)是具有非 0 对角线元素的方阵。
- [p,q,r] = dmperm(A)：A 为方阵，p 为行排列向量，q 为列排列向量，使得 A(p,q)是上三角块形式，r 为索引向量。
- [p,q,r,s] = dmperm(A)：A 不是方阵，p,q,r 含义与前面相同，s 也是索引向量。

【功能介绍】 返回矩阵的行排列向量。

【实例 2.135】 求得矩阵 $A=\begin{bmatrix}9&0&43&0\\61&5&0&44\\3&22&0&24\\0&9&63&2\end{bmatrix}$ 的行排列向量。

```
>> A=[9 0 43 0;61 5 0 44;3 22 0 24;0 9 63 2]
A =
     9     0    43     0
    61     5     0    44
     3    22     0    24
     0     9    63     2
>> [p,q,r]=dmperm(A)
p =
     4     3     2     1
q =
     3     4     2     1
r =
     1     5
>> a=A(p,q)
a =
    63     2     9     0
     0    24    22     3
     0    44     5    61
    43     0     0     9
```

2.7.18 randperm 函数——整数的随机排列

【语法说明】 p = randperm (n) ：对正整数 1，2，3，…，n 的随机排列，可用来创建随机变换矩阵。

【实例 2.136】 对 9 个整数进行随机排列。

```
>> p=randperm(9)
p =
     8     2     7     4     3     6     9     5     1
```

2.7.19 condest 函数——稀疏矩阵的 1-范数

【语法说明】

- c = condest(A) ：计算方阵 A 的 1-范数中条件数的下界值 c。

▣ [c,v] = condest(A)：方阵 A 的 1-范数中条件数的下界值 c 和向量 v，使得$\| Av \| = (\| A \| \cdot \| v \|)/c$，即 norm(A*v,1) = norm(A,1)*norm(v,1)/c。

【功能介绍】 矩阵 A 条件数的 1-范数估计值。

【实例 2.137】 求矩阵 A 条件数的 1-范数估计值。

```
>> A=[1,2,3;4,5,6;7,8,9]
A =
     1     2     3
     4     5     6
     7     8     9
>> c = condest(A)
c =
   4.5392e+017
>> [c,v] = condest(A)
c =
   4.5392e+017
v =
   -0.2500
    0.5000
   -0.2500
```

【实例讲解】 求得的 *c* 为方阵 ***A*** 的 1-范数的条件数的下限。

2.7.20 normest 函数——稀疏矩阵的 2-范数估计值

【语法说明】

▣ nrm = normest(S)：返回矩阵 S 的 2-范数的估计值，相对误差为 10^{-6}。

▣ nrm = normest(S,tol)：tol 为指定的相对误差，而不用默认误差 10^{-6}。

▣ [nrm,count] = normest(…)：count 为给出的计算范数迭代的次数。

【功能介绍】 求稀疏矩阵的 2-范数估计值。

【实例 2.138】 创建一个新的稀疏矩阵，并对该矩阵求 2-范数。

```
>> S=sparse(1:10,1:10,1:10)
S =
```

```
   (1,1)        1
   (2,2)        2
   (3,3)        3
   (4,4)        4
   (5,5)        5
   (6,6)        6
   (7,7)        7
   (8,8)        8
   (9,9)        9
  (10,10)      10
>> N1 = normest(S)
N1 =
   10.0000
>>  N2 = normest(S,'1e-6')
N2 =
   19.6214
```

2.7.21 luinc 函数——稀疏矩阵的分解

【语法说明】

- [L,U] = luinc(X,'0') ：X 为稀疏方阵，L 为下三角矩阵的置换形式，U 为上三角矩阵，0'是一种分解标准。
- [L,U,P] = luinc(X,'0') ：L 为下三角阵，其主对角线元素为 1；U 为上三角阵，p 为单位矩阵的置换形式。
- [L,U] = luinc(X,options) ：options 取值为 droptol 表示指定的舍入误差，为 milu 表示改变分解以便于从上三角分解因子中抽取被去掉的列元素。ugiag 为 1 表示用 droptol 值代替上三角因子中的对角线上的零元素，默认值为 0。thresh 为中心临界值。
- [L,U] = luinc(X,droptol) ：droptol 表示指定不完全分解的舍入误差。
- [L,U,P] = luinc(X,options)：options 取值为：droptol 表示指定的舍入误差；milu 表示改变分解以便于从上三角分解因子中抽取被去掉的列元素。ugiag 为 1 表示用 droptol 值代替上三角因子中的对角线上的零元素，默认值为 0。thresh 为中心临界值。

[L,U,P] = luinc(X,droptol)：droptol 表示确定不完全分解的舍入误差。

【功能介绍】 不完全的 LU 分解。

【实例 2.139】 对稀疏矩阵 $\boldsymbol{A}=\begin{bmatrix}11 & 0 & 13 & 0\\41 & 0 & 0 & 44\\0 & 22 & 0 & 24\\0 & 0 & 63 & 0\end{bmatrix}$ 进行不完全分解。

```
>> S=[11 0 13 0;41 0 0 44;0 22 0 24;0 0 63 0]
S =
    11     0    13     0
    41     0     0    44
     0    22     0    24
     0     0    63     0
>> S=sparse(S)
S =
   (1,1)       11
   (2,1)       41
   (3,2)       22
   (1,3)       13
   (4,3)       63
   (2,4)       44
   (3,4)       24
>> luinc(S,'0')
ans =
   (1,1)      41.0000
   (4,1)       0.2683
   (2,2)      22.0000
   (3,3)      63.0000
   (4,3)       0.2063
   (1,4)      44.0000
   (2,4)      24.0000
>> [L,U,p]=luinc(S,'0')
L =
   (1,1)       1.0000
```

```
        (4,1)         0.2683
        (2,2)         1.0000
        (3,3)         1.0000
        (4,3)         0.2063
        (4,4)         1.0000
   U =
        (1,1)         41
        (2,2)         22
        (3,3)         63
        (1,4)         44
        (2,4)         24
   p =
        (4,1)          1
        (1,2)          1
        (2,3)          1
        (3,4)          1
```

【实例讲解】 稀疏矩阵的 LU 分解对于求解稀疏矩阵是十分重要的方法。

2.7.22 eigs 函数——稀疏矩阵的特征值分解

【语法说明】

- d = eigs(A)：求稀疏矩阵 A 的 6 个最大特征值 d，d 以向量形式存放。
- d = eigs(A,B) ：求稀疏矩阵的广义特征值问题。满足 AV=BVD，其中 D 为特征值对角阵，V 为特征向量矩阵，B 必须是对称正定阵或 Hermitian 正定阵。
- d = eigs(A,k) ：返回 k 个最大特征值。
- d = eigs(A,B,k)：返回 k 个最大特征值。
- d = eigs(A,k,sigma) ：sigma 取值为 lm 表示最大数量的特征值为 sm 表示最小数量特征值；对实对称问题，la 表示最大特征值，sa 为最小特征值；对非对称和复数问题 lr 表示最大实部，sr 表示最小实部，li 表示最大虚部； si 表示最小虚部。

- d = eigs(A,B,k,sigma)：同上。
- d = eigs(A,k,sigma,options)：options 为指定参数，参见 eigs 帮助文件。
- d = eigs(A,B,k,sigma,options) ：同上。
- d = eigs(Afun,n) ：用函数 Afun 代替 A，n 为 A 的阶数，D 为特征值。
- [V,D] = eigs(A,⋯)：D 为 6 个最大特征值对角阵，V 的列向量为对应特征向量。
- [V,D,flag] = eigs(A,⋯)：flag 表示特征值的收敛性，若 flag=0，则所有特征值都收敛，否则，不是所有都收敛。

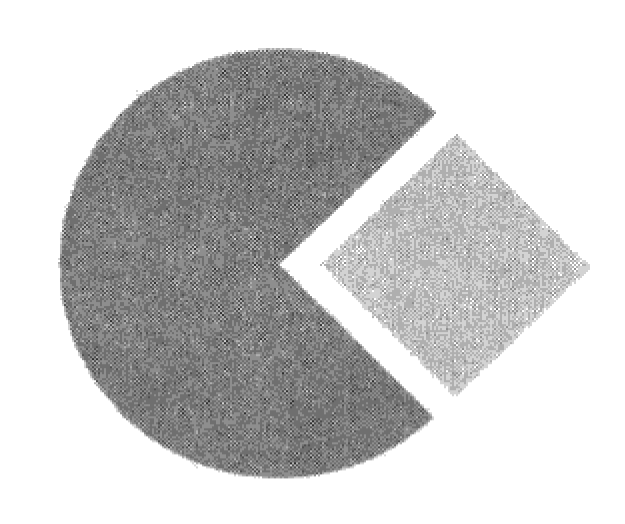
【功能介绍】 对稀疏矩阵进行特征值分解。

【实例 2.140】 对矩阵 $\boldsymbol{A}=\begin{bmatrix}23 & 24 & 34 & 7\\ 2 & 4 & 7 & 8\\ 45 & 89 & 36 & 78\\ 90 & 30 & 29 & 88\end{bmatrix}$进行特征值分解。

```
>> A = [23,24,34,7;2,4,7,9;45,89,36,78;90,30,29,88]
A =
    23    24    34     7
     2     4     7     9
    45    89    36    78
    90    30    29    88
>> d = eigs(A)
d =
1.0e+002 *
   1.4765
   0.0760 - 0.2954i
   0.0760 + 0.2954i
  -0.1184
>> d = eigs(A,4)
d =
 1.0e+002 *
   1.4765
   0.0760 - 0.2954i
   0.0760 + 0.2954i
```

```
  -0.1184
>> d = eigs(A,3)
d =
  1.0e+002 *
   1.4765
   0.0760 - 0.2954i
   0.0760 + 0.2954i
>> d = eigs(A,2,'sm')
Iteration 1: a few Ritz values of the 4-by-4 matrix:
     0
     0
     0
d =
 -11.8443
   7.5972 -29.5399i
```

【实例讲解】 稀疏矩阵的特征值是十分重要的矩阵信息。



第3章　数值计算函数

数值计算无论在理论和工程上都是最常用的，也是最基本的运算，MATLAB 中提供了几乎全部的数值运算函数，下面将从基本数学函数，插值、拟合与查表函数及数值微积分函数进行讲述。

3.1　基本数学函数

基本的数学函数是数值计算函数中很重要的部分，包括三角函数、绝对值、指数函数、对数函数、排序、求极限和对复数的操作等。这些在数学计算中常见的运算 MATLAB 都会有相应的函数，它们为复杂的数学运算提供了方便。

3.1.1　sin 和 sinh 函数——正弦函数与双曲正弦函数

【语法说明】

- Y = sin(X)：计算参量 X（可以是向量、矩阵，元素可以是复数）中每一个角度分量的正弦值 Y，所有分量的角度单位为弧度。
- Y = sinh(X)：计算参量 X 的双曲正弦值 Y。

【功能介绍】 生成正弦函数和双曲正弦函数。

【实例 3.1】 绘制正弦函数和双曲正弦函数。

```
>> x = -pi:0.02:pi;
>> plot(x,sin(x))
>> x=-4:0.02:4;
>>plot(x,sinh(x))
```

【实例讲解】 这两个函数比较常用，生成的曲线如图 3.1 和图 3.2 所示。

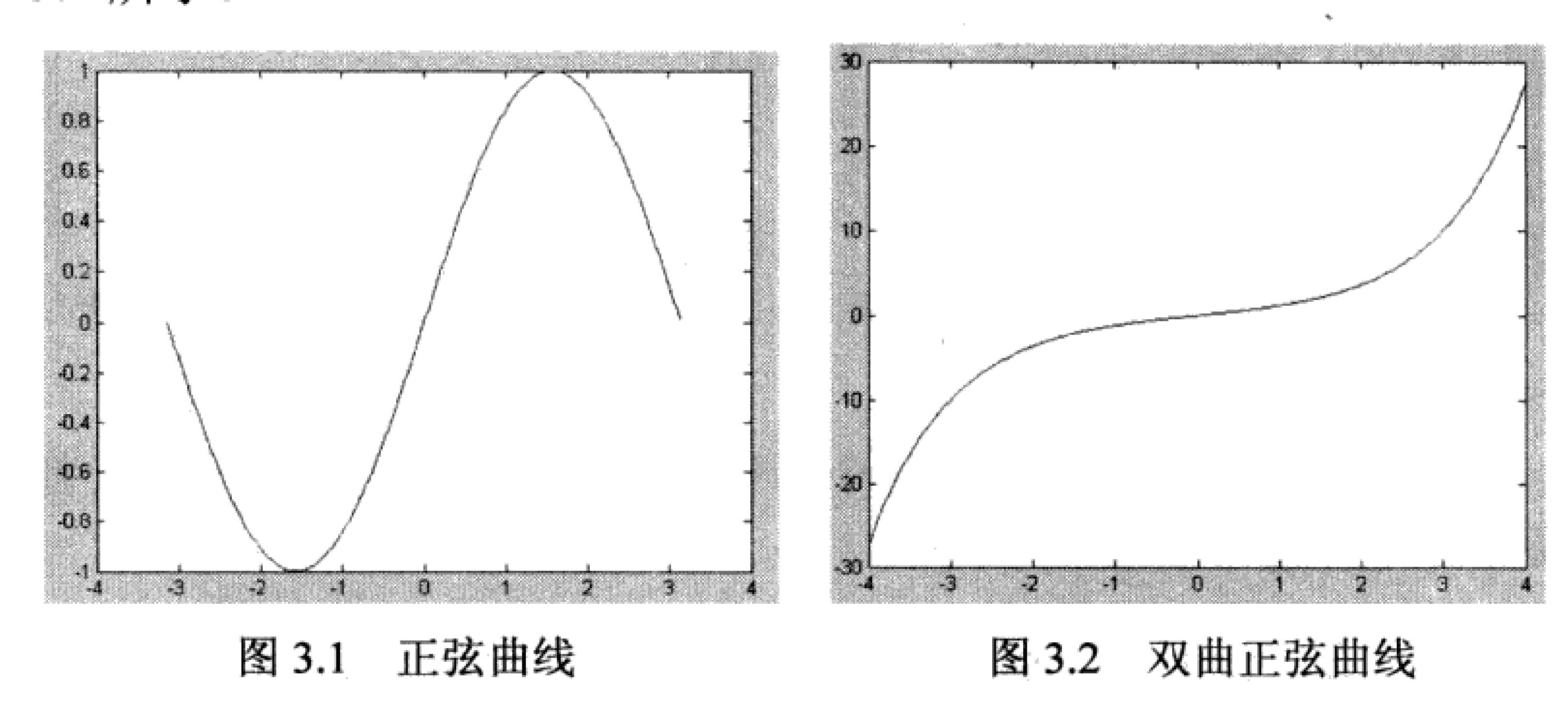


图 3.1 正弦曲线　　图 3.2 双曲正弦曲线

3.1.2 asin、asinh 函数——反正弦函数与反双曲正弦函数

【语法说明】

- Y = asin(X)：返回参量 X（可以是向量、矩阵）中每一个元素的反正弦函数值 Y。若 X 中有的分量处于 [−1,1] 之间，则 Y = asin(X)对应的分量处于[−π/2,π/2]之间；若 X 中有分量在区间[−1,1]之外，则 Y= asin(X)对应的分量为复数。
- Y = asinh(X)：返回参量 X 中每一个元素的反双曲正弦函数值。

【功能介绍】 求反正弦函数和反双曲正弦函数。

【实例 3.2】 画出反正弦函数和反双曲正弦函数。

```
x = -1:.01:1;
 plot(x,asin(x))
x = -5:.01:5;
plot(x,asinh(x))
```

上面实例得到的结果如图 3.3 和图 3.4 所示。

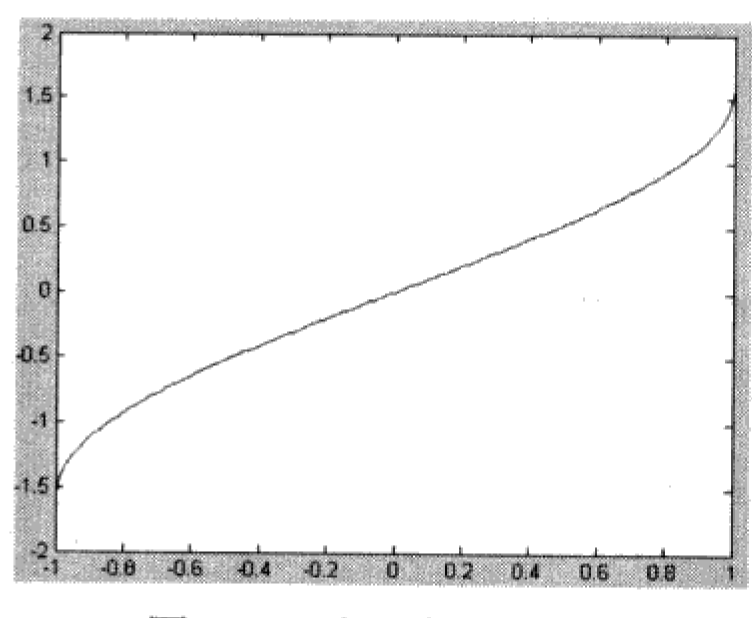


图 3.3　反正弦函数图

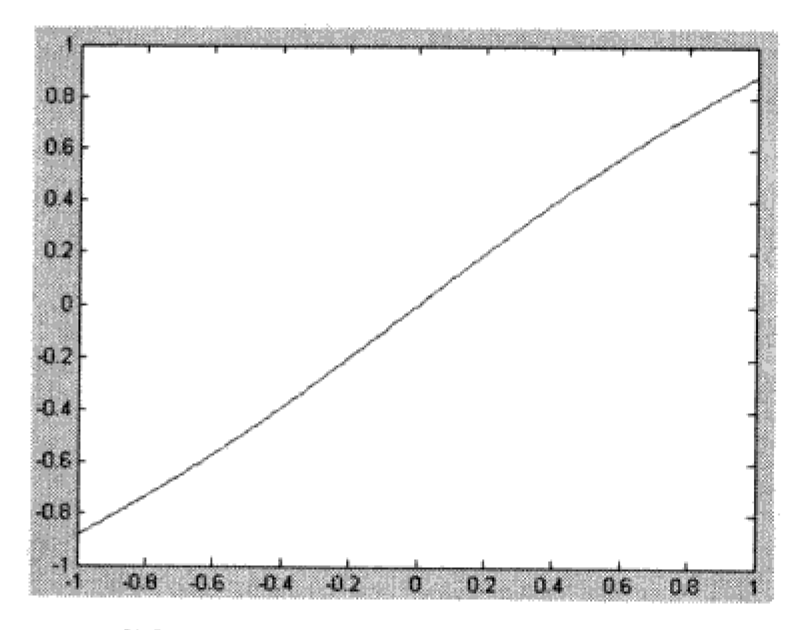


图 3.4　反双曲正弦函数图

【实例讲解】 通过上面两个语句可以看出，绘制函数的步长都是 0.1，但是两者的区间不同。

3.1.3　cos、cosh 函数——余弦函数与双曲余弦函数

【语法说明】

- Y = cos(X)：计算参量 X（可以是向量、矩阵，元素可以是复数）中每一个角度分量的余弦值 Y，所有角度分量的单位为弧度。我们要指出的是，cos(pi/2)并不是精确的零，而是与浮点精度有关的无穷小量 eps，因为 pi 仅仅是精确值π浮点近似的表示值而已。
- Y = cosh(X)：计算参量 X 的双曲余弦值 Y。

【功能介绍】 求取余弦函数与双曲余弦函数。若 X 为复数 $z=x+iy$，则函数定义为：$\cos(x+iy) = \cos(x)*\cos(y) + i*\sin(x)*\sin(y)$，$\cos z=\dfrac{e^{iz}+e^{-iz}}{2}$，$\cosh z=\dfrac{e^{z}+e^{-z}}{2}$

【实例 3.3】 绘制从$\left[-\pi,\pi\right]$上的余弦函数和反余弦函数。

```
>> x = -pi:0.01:pi;
>>plot(x,cos(x))
>>plot(x,cosh(x))
```

上面实例得到的结果如图 3.5 和图 3.6 所示。

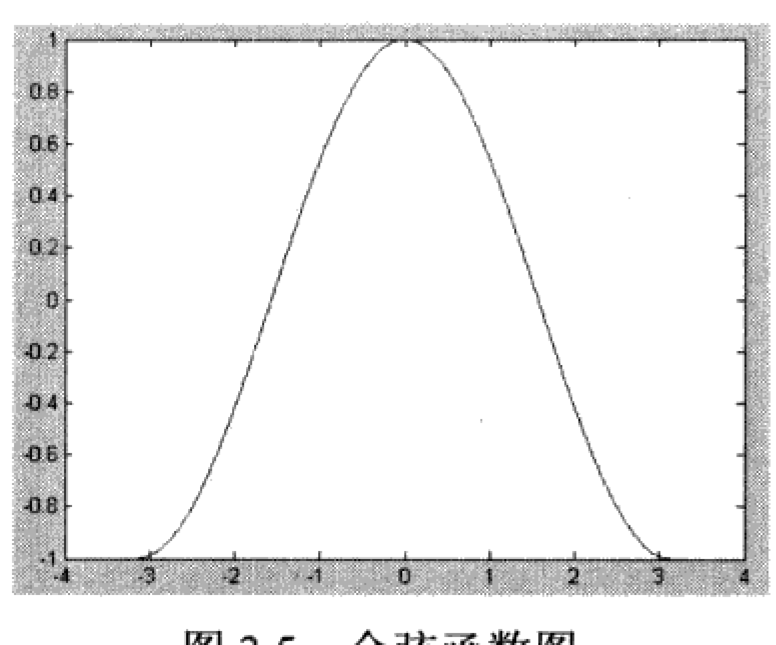


图 3.5　余弦函数图

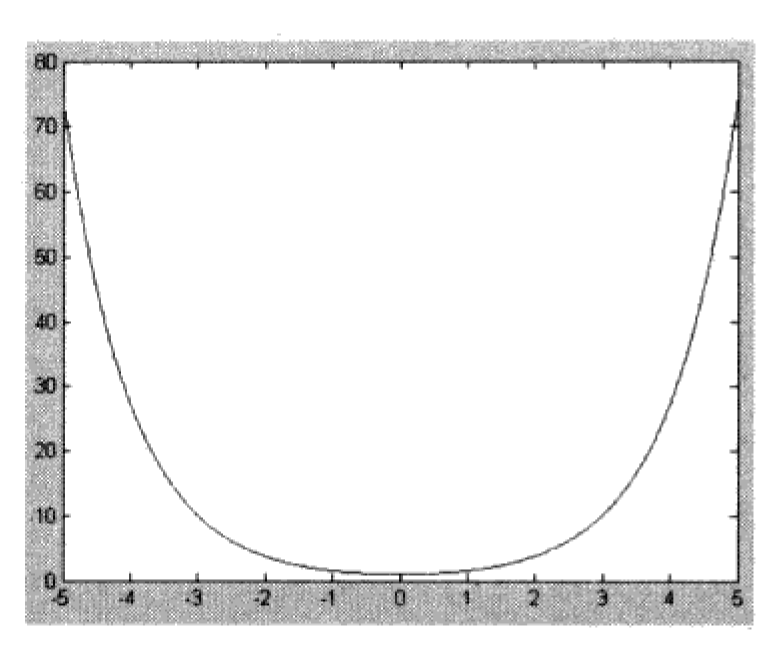


图 3.6　双曲余弦函数图

【实例讲解】 用户可以在同一个窗口中绘制上面两个函数的图形。

3.1.4　acos、acosh 函数——反余弦函数与反双曲余弦函数

【语法说明】

- Y = acos(X)：返回参量 X（可以是向量、矩阵）中每一个元素的反余弦函数值 Y。若 X 中有的分量处于 [−1,1] 之间，则 Y = acos(X)对应的分量处于 [0, π] 之间，若 X 中有分量在区间 [−1,1] 之外，则 Y = acos(X)对应的分量为复数。
- Y = acosh (X)：返回参量 X 中每一个元素的反双曲余弦函数 Y。

【语法说明】 求出反余弦函数与反双曲余弦函数。

【实例 3.4】 画出 $[-1,1]$ 区间上的反余弦函数和反双曲余弦函数。

```
>>x = -1:.01:1;
>>plot(x,acos(x))
>>x = -5:.01:5;
>>plot(x,acosh(x))
```

上面实例得到的结果如图 3.7 和图 3.8 所示。

【实例讲解】 x 指定了区间和步长，绘制曲线的时候以 x 为横

坐标，分别以 x 的反余弦函数和反双曲余弦函数为纵坐标。

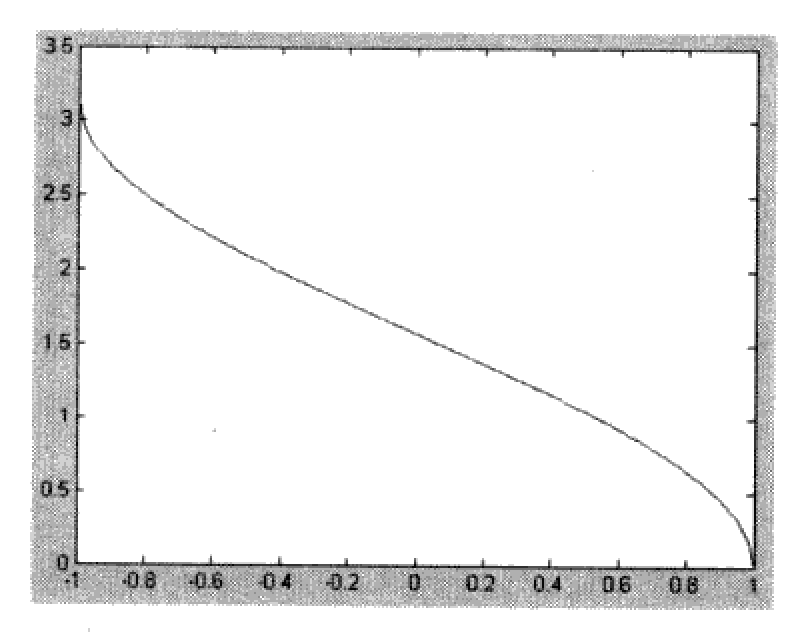


图 3.7　反余弦函数图

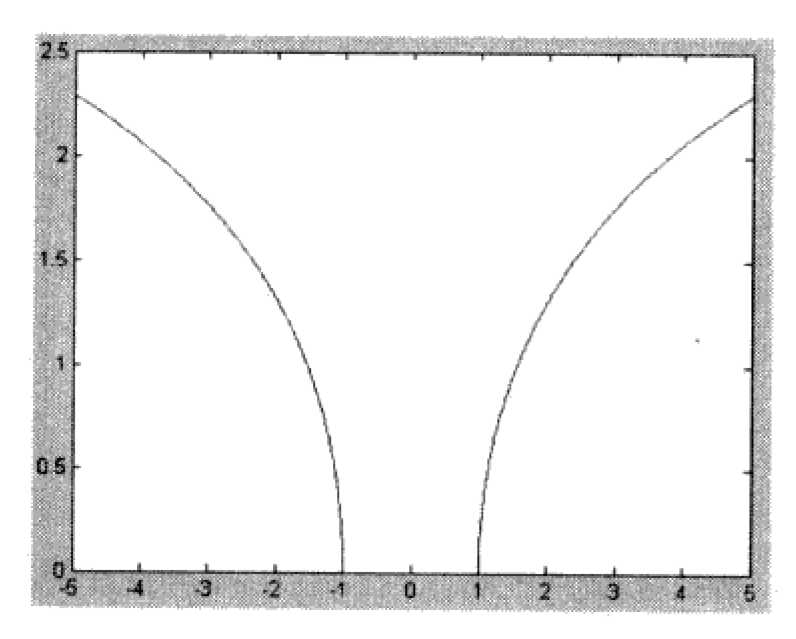


图 3.8　反双曲余弦函数图

3.1.5　tan 和 tanh 函数——正切函数与双曲正切函数

【语法说明】

□ Y = tan(X)：计算参量 X（可以是向量、矩阵，元素可以是复数）中每一个角度分量的正切值 Y，所有角度分量的单位为弧度。我们要指出的是，tan(pi/2)并不是精确的零，而是与浮点精度有关的无穷小量 eps，因为 pi 仅仅是精确值π浮点近似的表示值而已。

□ Y = tanh(X)：返回参量 X 中每一个元素的双曲正切函数值 Y。

【功能介绍】 求正切函数与双曲正切函数。

【实例 3.5】 画出 $\left[-\frac{\pi}{2},\frac{\pi}{2}\right]$ 上的正切和双曲正切函数。

```
>>= (-pi/2)+0.01:.01:(pi/2) -0.01;% 去除奇异点
>>plot(x,tan(x))
>> -5:0.01:5;
>> plot(x,tanh(x))
```

得到的结果如图 3.9 和图 3.10 所示。

【实例讲解】 x 指定了区间和步长，绘制曲线的时候以 x 为横

坐标，分别以 x 的正切函数与双曲正切函数为纵坐标，在奇异点处要分别增减一个小数。

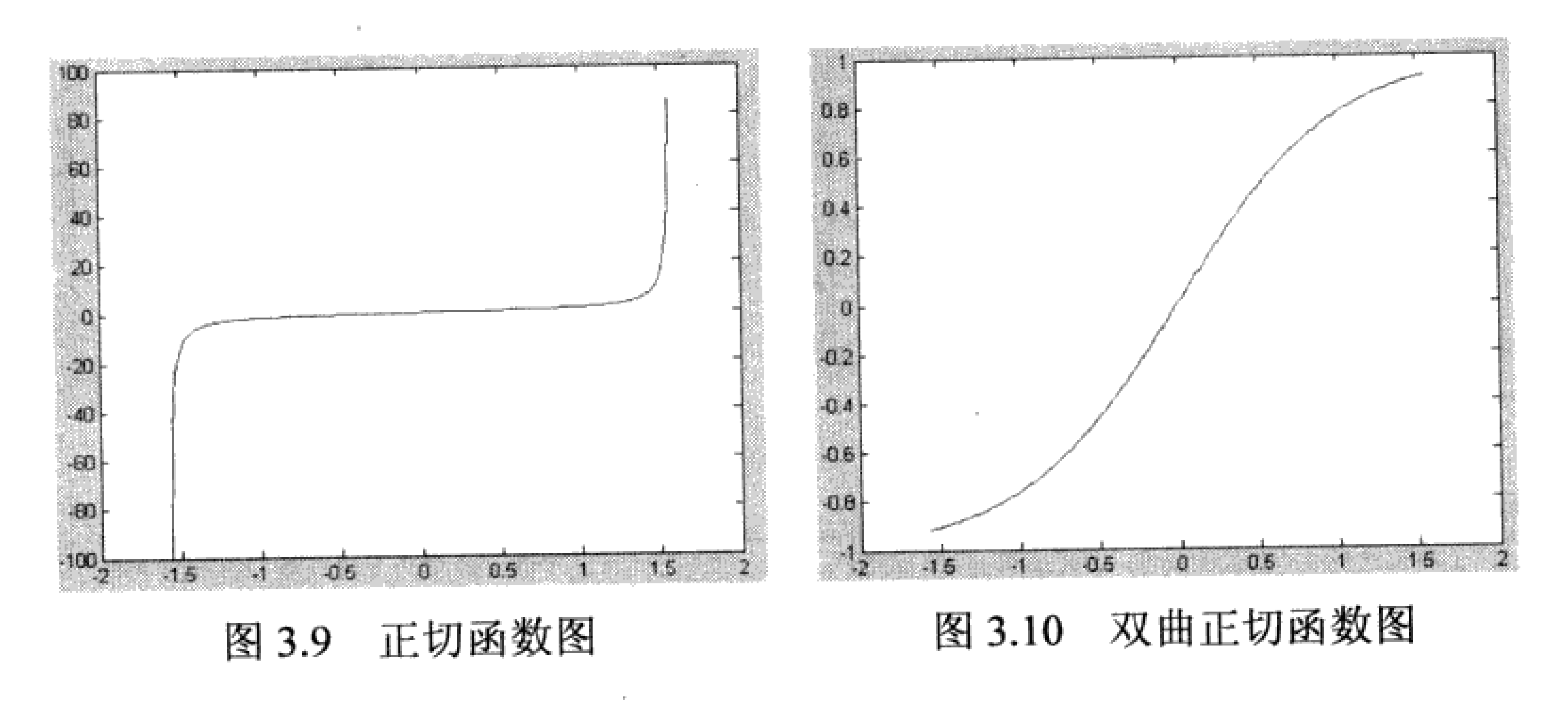


图 3.9　正切函数图　　　　图 3.10　双曲正切函数图

3.1.6　atan、atanh 函数——反正切函数与反双曲正切函数

【语法说明】

- Y = atan(X)：返回参量 X（可以是向量、矩阵）中每一个元素的反正切函数值 Y。若 X 中有的分量为实数，则 Y = atan(X) 对应的分量处于[$-\pi/2,\pi/2$]之间。
- Y = atanh(X)：返回参量 X 中每一个元素的反双曲正切函数值 Y。

【功能介绍】 求取反正切函数与反双曲正切函数。

【实例 3.6】 画出反正切函数与反双曲正切函数。

```
>>x = -20:0.01:20;
>>plot(x,atan(x))
>>x = -1:0.01:1;
>>plot(x,atanh(x))
```

得到的结果如图 3.11 和图 3.12 所示。

【实例讲解】 x 指定了区间和步长，绘制曲线的时候以 x 为横坐标，分别以 x 的反正切函数与反双曲正切函数为纵坐标。

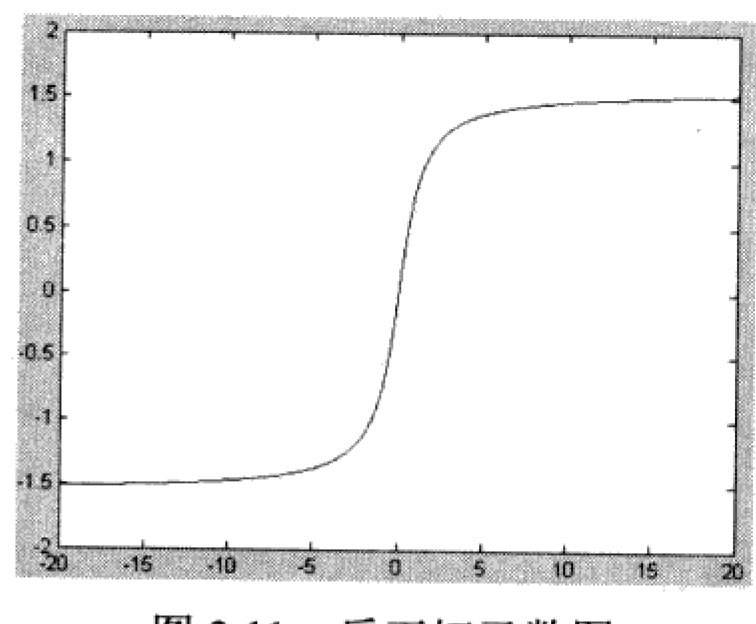


图 3.11 反正切函数图

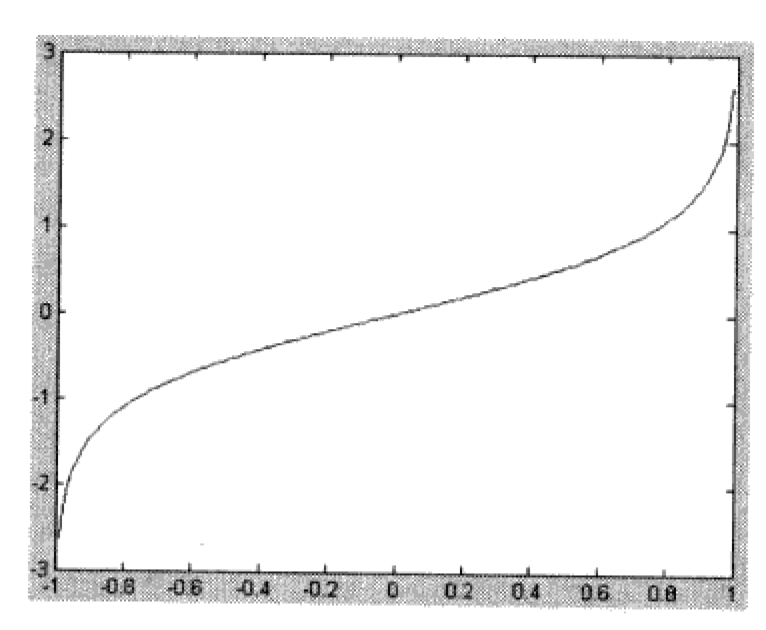


图 3.12 反双曲正切函数

3.1.7 cot、coth 函数——余切函数与双曲余切函数

【语法说明】

- Y = cot(X)：计算参量 X（可以是向量、矩阵，元素可以是复数）中每一个角度分量的余切值 Y，所有角度分量的单位为弧度。
- Y = coth(X)：返回参量 X 中每一个元素的双曲余切函数值 Y。

【功能介绍】 求给定自变量范围的余切函数与双曲余切函数。

【实例 3.7】 画出余切函数与双曲余切函数。

```
>>x1 = -pi+0.01:0.01: -0.01;  %  去掉奇点 x = 0
>>x2 = 0.01:0.01:pi-0.01;  % 同上
>>plot(x1,cot(x1),x2,cot(x2))
>>plot(x1,coth(x1),x2,coth(x2))
```

得到的结果如图 3.13 和图 3.14 所示。

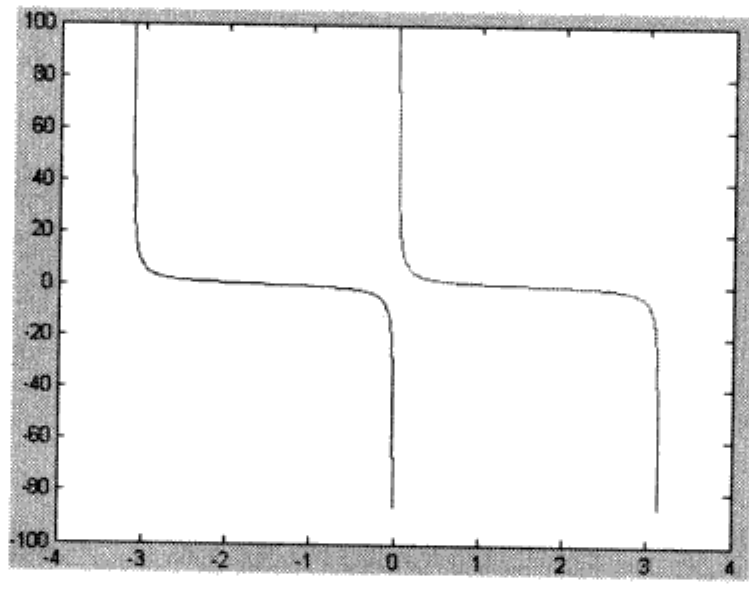


图 3.13 余切函数图

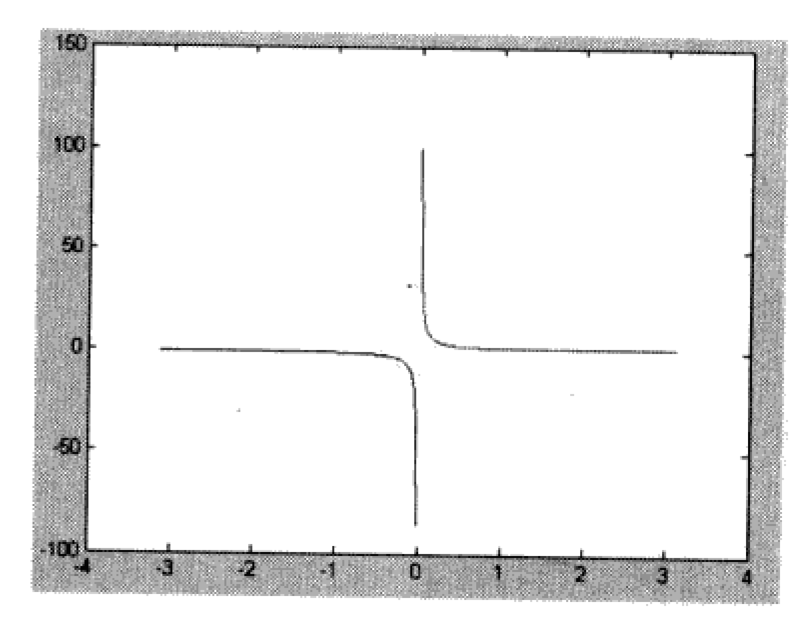


图 3.14 双曲余切函数

【实例讲解】 在这个例子中，注意将奇点（函数值为无穷大或无穷小的点）位置先挖除。

3.1.8　acot、acoth 函数——反余切函数与反双曲余切函数

【语法说明】

- Y = acot(X)：返回参量 X（可以是向量、矩阵）中每一个元素的反余切函数 Y。
- Y = acoth(X)：返回参量 X 中每一个元素的反双曲余切函数值 Y。

【功能介绍】 求反余弦函数与反双曲余切函数。

【实例 3.8】 绘制反余切函数与反双曲余切函数的图形。

```
>>x1 = -2*pi:pi/20: -0.99; x2 = 0.99:pi/20:2*pi; % 去掉奇异点 x = 0
>>plot(x1,acot(x1),x2,acot(x2))
>>x1 = -24:0.1: -1; x2 = 1:0.1:24;
>>plot(x1,acoth(x1),x2,acoth(x2))
```

得到的结果如图 3.15 和图 3.16 所示。

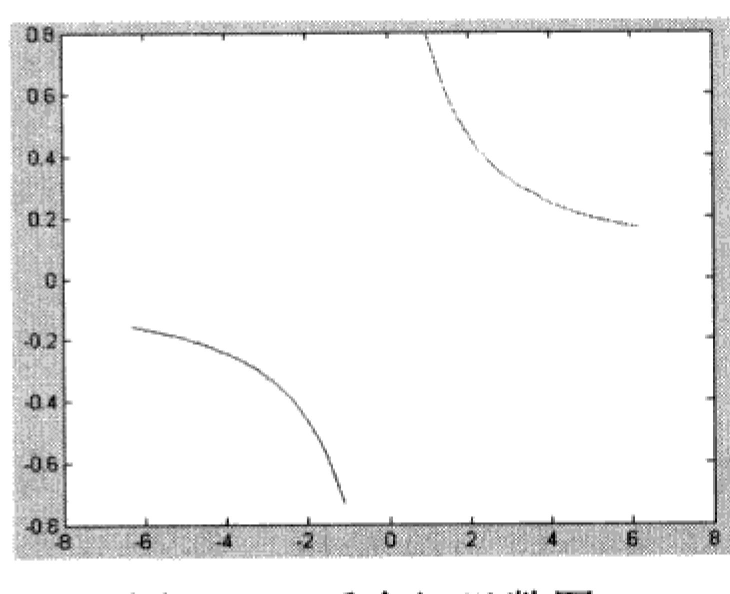


图 3.15　反余切函数图

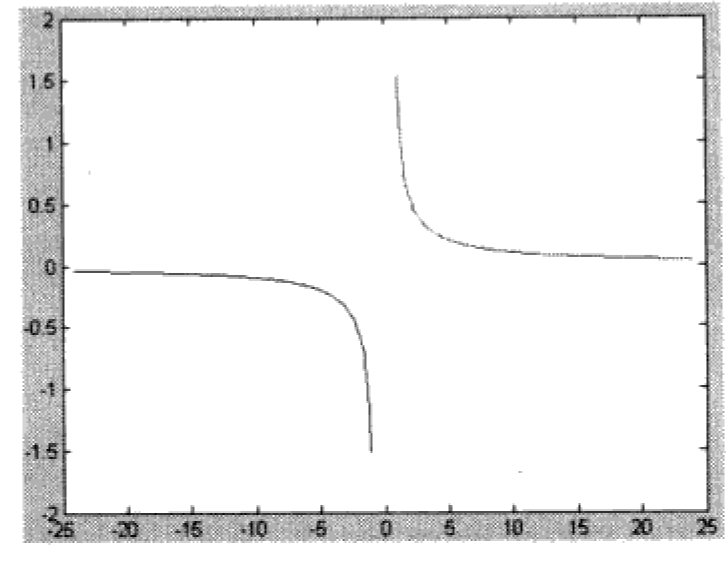


图 3.16　反双曲余切函数图

【实例讲解】 注意要去掉奇异点。

3.1.9　sec、sech 函数——正割函数与双曲正割函数

【语法说明】

- Y = sec(X)：计算参量 X（可以是向量、矩阵，元素可以是复

数）中每一个角度分量的正割函数值 Y，所有角度分量的单位为弧度。我们要指出的是，sec(pi/2)并不是无穷大，而是与浮点精度有关的无穷小量 eps 的倒数，因为 pi 仅仅是精确值π浮点近似的表示值而已。

- Y = sech(X)：返回参量 X 中每一个元素的双曲正割函数值 Y。

【功能介绍】 求正割函数与双曲正割函数。

【实例 3.9】 在$\left[-\frac{\pi}{2},\frac{\pi}{2}\right]$内绘制正割函数与双曲正割曲线。

```
>>x1 = -pi/2+0.01:0.01:pi/2-0.01;  % 去掉奇异点 x = pi/2
>>x2 = pi/2+0.01:0.01:(3*pi/2) -0.01;
>>plot(x1,sec(x1),x2,sec(x2))
>>x = -2*pi:0.01:2*pi;
>>plot(x,sech(x))
```

得到的结果如图 3.17 和图 3.18 所示。

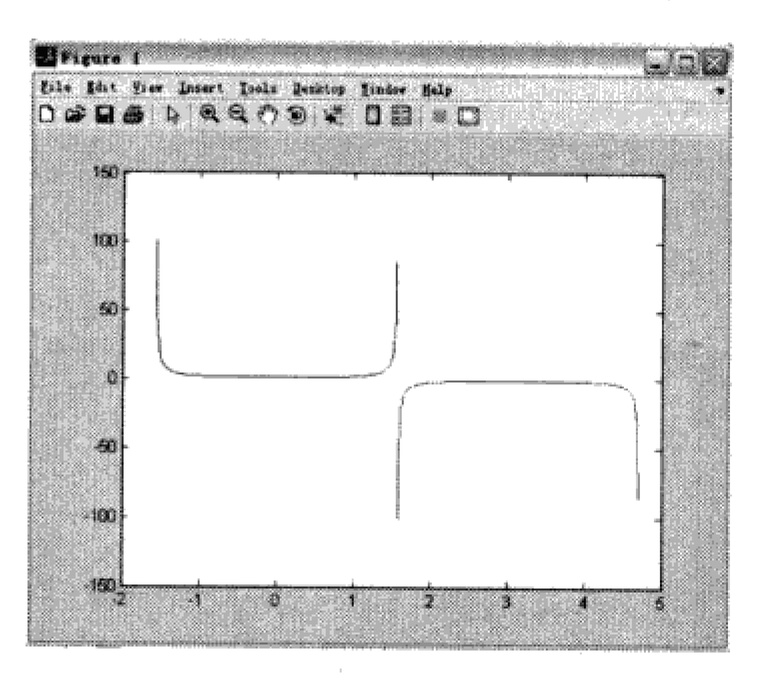


图 3.17 正割函数图

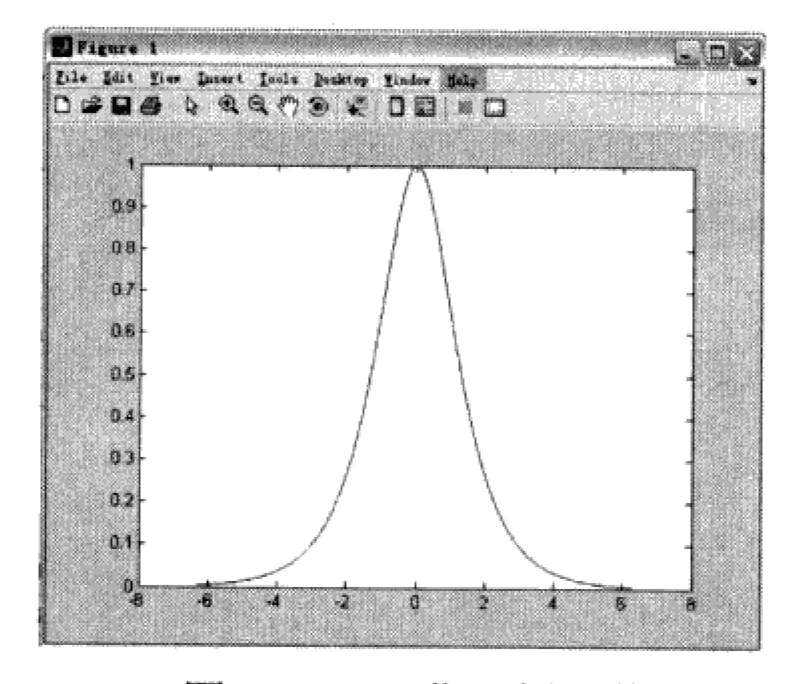


图 3.18 双曲正割函数图

【实例讲解】 x 指定了区间和步长，绘制曲线的时候以 x 为横坐标，分别以 x 的正割函数与双曲正割函数为纵坐标，在奇异点处要分别增减一个小数。

3.1.10 asec、asech 函数——反正割函数与反双曲正割函数

【语法说明】

- Y = asec(X)：返回参量 X（可以是向量、矩阵）中每一个

元素的反正割函数值 Y。

- Y = asech(X)：返回参量 X 中每一个元素的反双曲正割函数值 Y。

【功能介绍】 求取反正割函数与反双曲正割函数。

【实例 3.10】 画出反正割曲线与反双曲正割曲线。

```
>>x1 = -5:0.01: -1; x2 = 1:0.01:5;
>>plot(x1,asec(x1),x2,asec(x2))
>>x = 0.01:0.001:1;
>>plot(x,asech(x))
```

得到的结果如图 3.19 和图 3.20 所示。

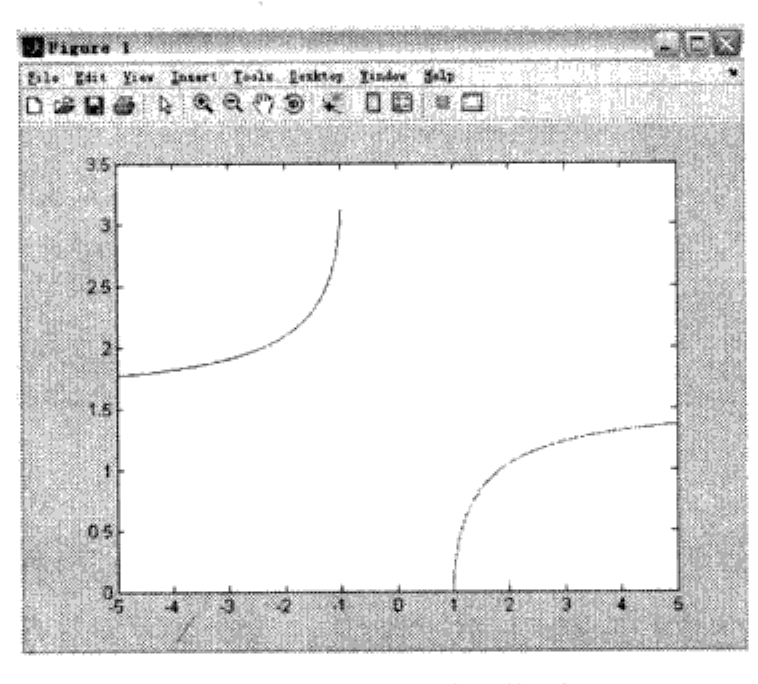


图 3.19　反正割曲线

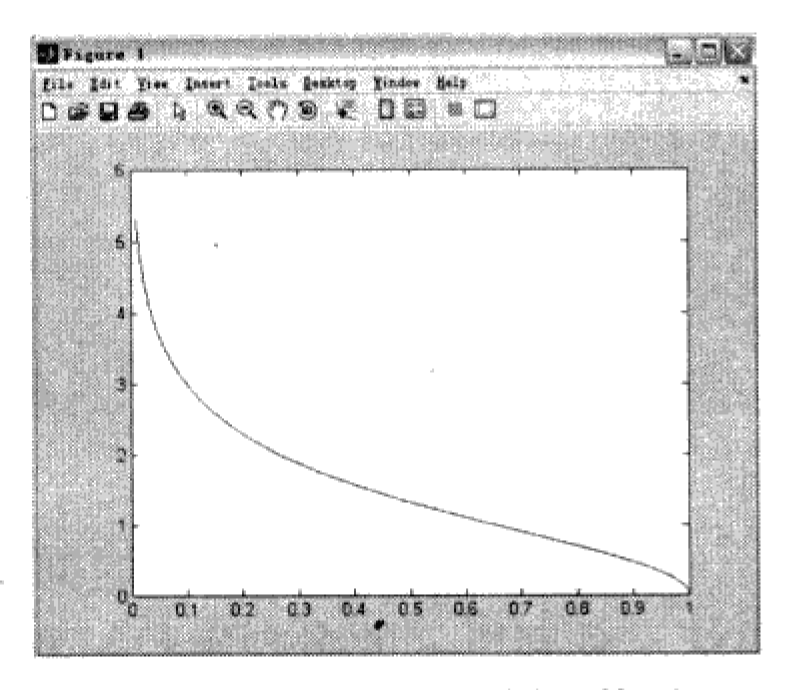


图 3.20　反双曲正割函数图

【实例讲解】 *x* 指定了区间和步长，绘制曲线的时候以 *x* 为横坐标，分别以 *x* 的反正割函数与反双曲正割函数为纵坐标，在奇异点处要分别增减一个小数。

3.1.11　csc、csch 函数——余割函数与双曲余割函数

【语法说明】

- Y = csc(X)：计算参量 X（可以是向量、矩阵，元素可以是复数）中每一个角度分量的余割函数值 Y，所有角度分量的单位为弧度。
- Y = csch(X)：返回参量 X 中每一个元素的双曲余割函数值 Y。

【功能介绍】 求取余割函数与双曲余割函数。

【实例 3.11】 绘制余割函数曲线与反余割函数曲线。

```
>>x1 = -pi+0.01:0.01: -0.01; x2 = 0.01:0.01:pi-0.01; %去掉奇异点 x=0
>>plot(x1,csc(x1),x2,csc(x2))
>>plot(x1,csch(x1),x2,csch(x2))
```

得到的结果如图 3.21 和图 3.22 所示。

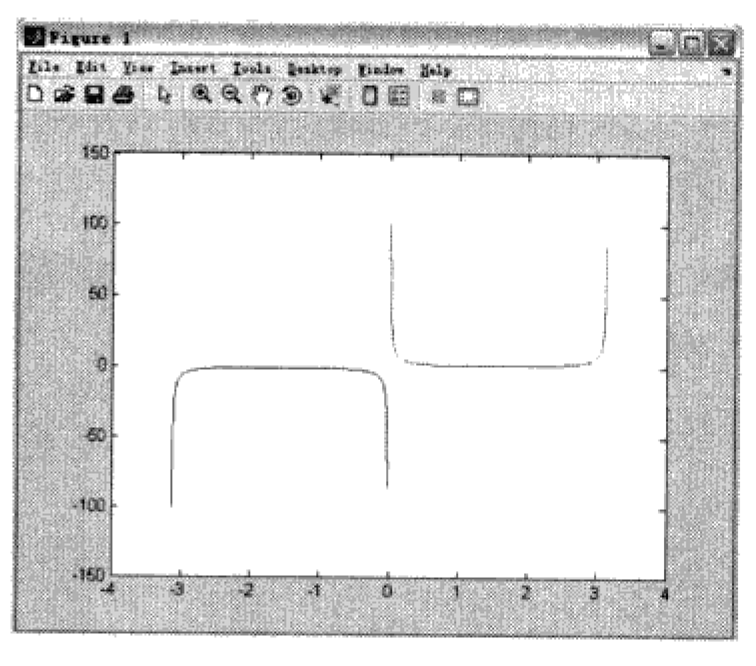


图 3.21 余割函数图

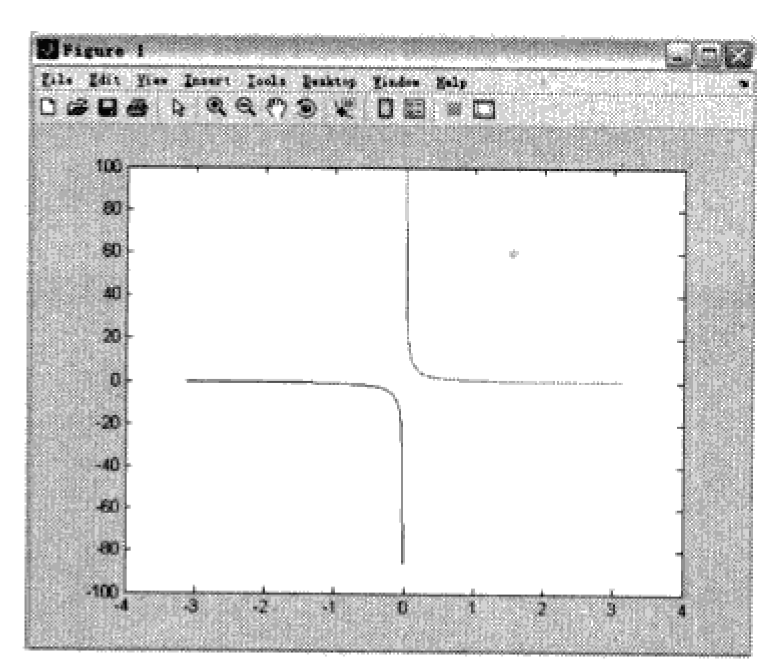


图 3.22 双曲余割函数图

【实例讲解】 x 指定了区间和步长，绘制曲线的时候以 x 为横坐标，分别以 x 的余割函数与双曲余割函数为纵坐标，在奇异点处要分别增减一个小数。

3.1.12 acsc、acsch 函数——反余割函数与反双曲余割函数

【语法说明】

- Y = acsc(X)：返回参量 X（可以是向量、矩阵）中每一个元素的反余割函数值 Y。
- Y = acsch(X)：返回参量 X 中每一个元素的反双曲余割函数值 Y。

【功能介绍】 求反余割函数与反双曲余割函数。

【实例 3.12】 绘制反余割曲线与反双曲余割函数曲线。

```
>>x1 = -10:0.01: -1.01; x2 = 1.01:0.01:10; % 去掉奇异点
x = 1
>>plot(x1,acsc(x1),x2,acsc(x2))
>>x1 = -20:0.01: -1; x2 = 1:0.01:20;
>>plot(x1,acsch(x1),x2,acsch(x2))
```

得到的结果如图 3.23 和图 3.24 所示。

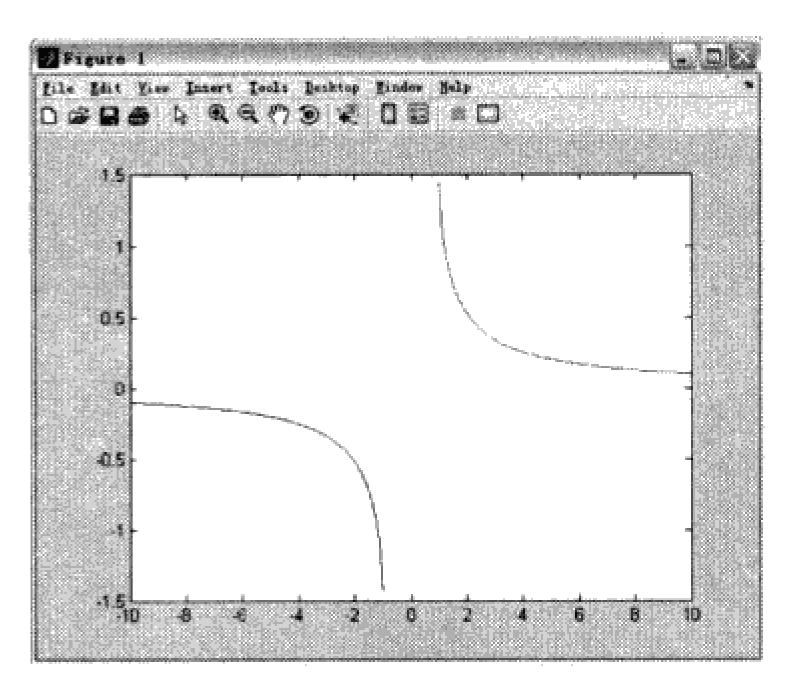


图 3.23 反余割函数图

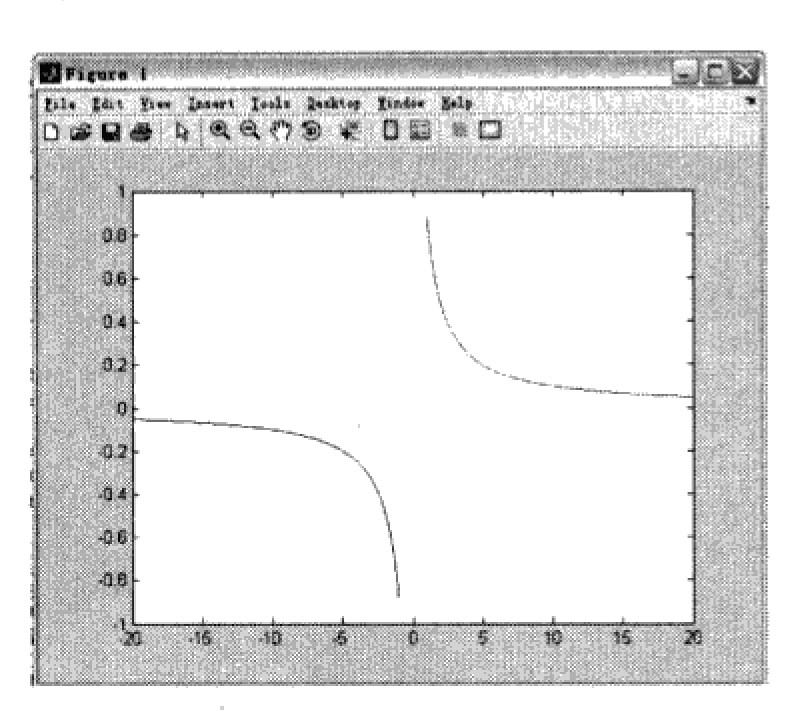


图 3.24 反双曲余割函数图

【实例讲解】 *x* 指定了区间和步长，绘制曲线的时候以 *x* 为横坐标，分别以 *x* 的反余割函数与反双曲余割函数为纵坐标。

3.1.13 atan2 函数——四象限的反正切函数

【语法说明】 P = atan2(Y,X)：返回一与参量 X 和 Y 同型的、与 X 和 Y 元素的实数部分对应的、元素对元素的四象限的反正切函数阵列 P，其中 X 和 Y 的虚数部分将忽略。阵列 P 中的元素分布在闭区间[−pi,pi]上。特定的象限将取决于 sign(Y)与 sign(X)。

【功能介绍】 求四象限表示的反正切函数。

【实例 3.13】 绘制四象限表示的反正切函数图。

```
>>z=3+10i;
>>r = abs(z);
>>anl = atan2(imag(z),real(z))
>>z = r *exp(i *anl)
>>feather(z);hold on
```

```
>>t=0:0.01:2*pi;
>>x=1+r*cos(t);
>>y=r*sin(t);
>>plot(x,y);
>>axis equal;
>>hold off
```

计算结果为：

```
anl =
    1.2793
z =
   3.0000 +10.0000i
```

得到的结果如图 3.25 所示。

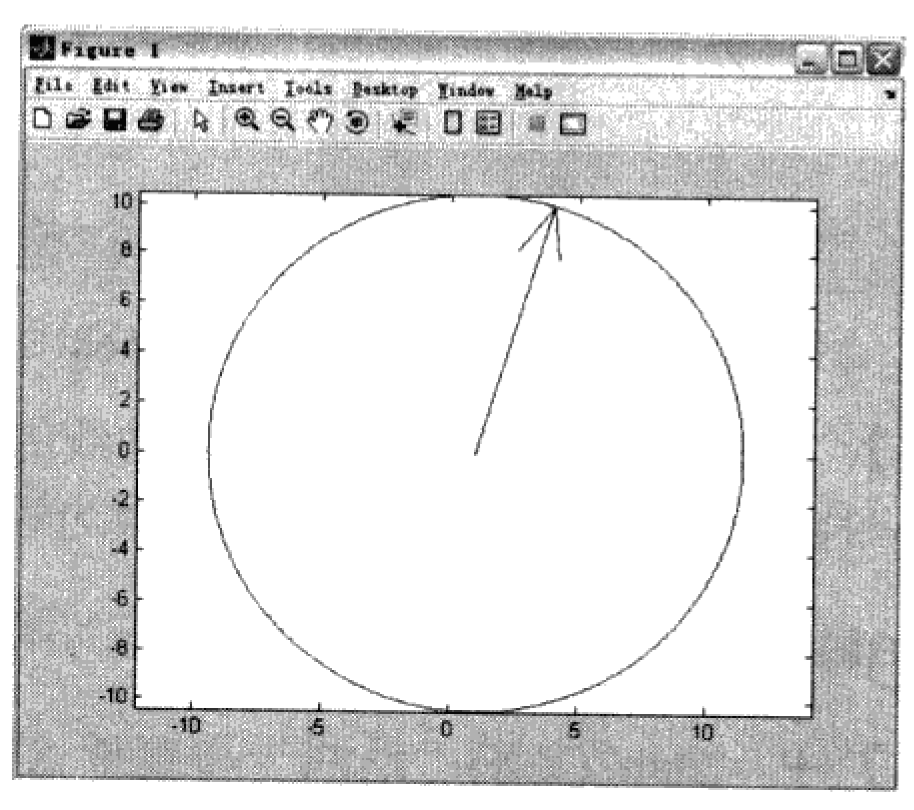


图 3.25　四象限的反正切函数图

【实例讲解】 首先对复数 z 求得其模长和角度，然后根据模长和角度求得随 t 变化的 x,y 坐标，而后进行绘图显示。

3.1.14　abs 函数——数值的绝对值与复数的幅值

【语法说明】 Y = abs(X)：返回参量 X 的每一个分量的绝对值；若 X 为复数，则返回每一分量的幅值 abs(X) = sqrt(real(X).^2+imag(X).^2)。

【功能介绍】 求实数或复数的绝对值。

【实例 3.14】 对向量 A = [−3.8, 0.3, 1.1415926, −0.8, 9.998,

2+6.6i]中的每个元素求绝对值。

```
>> A = [-3.8, 0.3, 1.1415926, -0.8, 9.998, 2+6.6i]
A =
  Columns 1 through 3
  -3.8000              0.3000              1.1416
  Columns 4 through 6
  -0.8000              9.9980              2.0000 + 6.6000i
>> Y = abs(A)
Y =
3.8000   0.3000   1.1416   0.8000   9.9980   6.8964
```

【实例讲解】 对于例子中的实数按照代数运算中的求绝对值进行，对于小数非常多的情况取小数点后 4 位。复数取其模值。

3.1.15 exp 函数——求以 e 为底的指数函数

【语法说明】 Y = exp(X)：对参量 X 的每一分量，求以 e 为底的指数函数 Y。X 中的分量可以为复数。对于复数分量，如 $z = x + i*y$，计算表达式为 e^z = e^x*(cos(y) + i*sin(y))。

【功能介绍】 求以 e 为底的指数函数。

【实例 3.15】 对向量 A = [−3.8, 0.3, 1.1415926, −0.8, 9.998, 2+6.6i]求以 e 为底的指数函数。

```
>> A = [-3.8, 0.3, 1.1415926, -0.8, 9.998, 2+6.6i];
>> Y = exp(A)
Y =
  1.0e+004 *
  Columns 1 through 3
   0.0000              0.0001              0.0003
  Columns 4 through 6
   0.0000              2.1982              0.0007 + 0.0002i
```

【实例讲解】 对向量求指数就是对向量中的每一个元素都求以 e 为底的指数函数。

3.1.16 expm 函数——求矩阵以 e 为底的指数函数

【语法说明】 Y = expm(X)：计算以 e 为底数、x 的每一个元素

为指数的指数函数值。若矩阵 x 有小于或等于零的特征值，则返回复数。

【功能介绍】 求矩阵以 e 为底的指数函数。

【实例 3.16】 用系统函数 hilb 生成一个 5×5 矩阵，然后求这个矩阵以 e 为底的指数函数。

```
>> A=hilb(5)
A =
    1.0000    0.5000    0.3333    0.2500    0.2000
    0.5000    0.3333    0.2500    0.2000    0.1667
    0.3333    0.2500    0.2000    0.1667    0.1429
    0.2500    0.2000    0.1667    0.1429    0.1250
    0.2000    0.1667    0.1429    0.1250    0.1111
>> Y = expm(A)
Y =
    3.3206    1.2579    0.8769    0.6768    0.5525
    1.2579    1.7774    0.5718    0.4543    0.3776
    0.8769    0.5718    1.4344    0.3524    0.2973
    0.6768    0.4543    0.3524    1.2904    0.2478
    0.5525    0.3776    0.2973    0.2478    1.2134
```

【实例讲解】 读者要仔细区分 exp 和 expm 函数的差别。

3.1.17 log 函数——求自然对数

【语法说明】 Y = log(X)：对参量 X 中的每一个元素计算自然对数。

【功能介绍】 求以 e 为底数的对数。

【实例 3.17】 对 ***A*** = [−3.8, 0.3, 1.1415926, −0.8, 9.998, 2+6.6i] 中的每个元素求自然对数。

```
>> A = [-3.8, 0.3, 1.1415926, -0.8, 9.998, 2+6.6i]
A =
  Columns 1 through 3
  -3.8000             0.3000             1.1416
  Columns 4 through 6
  -0.8000             9.9980             2.0000 + 6.6000i
```

```
>> X=log(A)
X =
  Columns 1 through 3
   1.3350 + 3.1416i  -1.2040             0.1324
  Columns 4 through 6
  -0.2231 + 3.1416i   2.3024             1.9310 + 1.2766i
```

【实例讲解】 如上例，A 中的元素可以是复数与负数，但由此可能得到意想不到的结果。若 $z = x + i \times y$，则 log 对复数的计算为 log (z) = log (abs (z)) + i*atan2(y,x)。

3.1.18　log10 函数——求常用对数

【语法说明】 Y = log10(X)：计算 X 中的每一个元素的常用对数。若 X 中出现复数，则可能得到意想不到的结果。

【功能介绍】 求以 10 为底数的对数。

【实例 3.18】 对 A = [−3.8, 0.3, 1.1415926, −0.8, 9.998, 2+6.6i]中的每个元素求常用对数。

```
>> A = [-3.8, 0.3, 1.1415926, -0.8, 9.998, 2+6.6i]
A =
  Columns 1 through 3
  -3.8000            0.3000             1.1416
  Columns 4 through 6
  -0.8000            9.9980             2.0000 + 6.6000i
>> X = log10(A)
X =
 Columns 1 through 3
   0.5798 + 1.3644i  -0.5229             0.0575
  Columns 4 through 6
  -0.0969 + 1.3644i   0.9999             0.8386 + 0.5544i
```

【实例讲解】 读者可以仔细对比矩阵中实数和复数计算结果的差异。

3.1.19　sort 函数——排序函数

【语法说明】

- B = sort(A)：沿着输入参量 A 的不同维的方向，从小到大

重新排列 A 中的元素。A 可以是字符串的、实数的、复数的单元数组。对于 A 中完全相同的元素，则按它们在 A 中的先后位置排列在一块；若 A 为复数的，则按元素幅值的从小到大排列，若有幅值相同的复数元素，则再按它们在区间[−π,π]的幅角从小到大排列。

- B = sort(A,dim)：沿着矩阵 A（向量的、矩阵的或多维的）中指定维数 dim 方向重新排列 A 中的元素。
- [B,INDEX] = sort(A,…)：输出参量 B 的结果如同上面的情形，输出 INDEX 是等于 size(A)的数组，它的每一列是与 A 中列向量的元素相对应的置换向量。

【功能介绍】 把输入参量中的元素按从小到大的方向重新排列。

【实例 3.19】 对 A = [−3.8, 0.3, 1.1415926, −0.8, 9.998, 2+6.6i]中的元素排序。

```
>> A = [-3.8, 0.3, 1.1415926, -0.8, 9.998, 2+6.6i];
>> B = sort(A)
B =
  Columns 1 through 3
   0.3000            -0.8000              1.1416
  Columns 4 through 6
  -3.8000             2.0000 + 6.6000i   9.9980
>> [B1,INDEX] = sort(A)
B1 =
  Columns 1 through 3
   0.3000            -0.8000              1.1416
  Columns 4 through 6
  -3.8000             2.0000 + 6.6000i   9.9980
INDEX =
     2     4     3     1     6     5
```

【实例讲解】 INDEX 向量标示了重新排列后元素在原向量中的位置。

3.1.20 fix 函数——向零方向取整

【语法说明】 B = fix(A)：对 A 的每一个元素朝零的方向取整数

部分，返回与 A 同维的数组。对于复数参量 A，则返回一复数，其分量的实数与虚数部分分别取原复数的、朝零方向的整数部分。

【功能介绍】 对数值向零方向取整。

【实例 3.20】 对向量 A=[−3.8, 0.3, 1.1415926, −0.8, 9.998, 2+6.6i]向零方向取整。

```
>> A = [-3.8, 0.3, 1.1415926, -0.8, 9.998, 2+6.6i];
>> X=fix(A)
X =
  Columns 1 through 3
  -3.0000                0                1.0000
  Columns 4 through 6
       0          9.0000               2.0000 + 6.0000i
```

【实例讲解】 取整的意图就是将小数化为整数，而向零的方向相当于以零为基准，负分数或小数取整的结果是靠近零方向的值，也就是取较大的那个整数；正分数或小数取整的结果也是靠近零方向的值，取较小的那个整数。

3.1.21　roud 函数——朝最近的方向取整

【语法说明】 Y = round(X)：对 X 的每一个元素朝最近的方向取整数部分，返回与 X 同维的数组；对于复数参量 X，则返回一复数，其分量的实数与虚数部分分别取原复数的、朝最近方向的整数部分。

【功能介绍】 对数值向最近的方向取整。

【实例 3.21】 将向量 A = [−1.9, −0.2, 3.1415926, 5.6, 7.0, 2.4+3.6i]中的数值向最近的方向取整。

```
>>A = [-1.9, -0.2, 3.1415926, 5.6, 7.0, 2.4+3.6i];
>>Y = round(A)
```

计算结果为：

```
Y =
     Columns 1 through 4
       -2.0000       0          3.0000            6.0000
     Columns 5 through 6
       7.0000      2.0000 + 4.0000i
```

【实例讲解】 即采用四舍五入的原则处理小数或分数部分。

3.1.22 floor 函数——朝负无穷大方向取整

【语法说明】 B = floor(A) ：对 A 的每一个元素朝负无穷大的方向取整数部分，返回与 A 同维的数组；对于复数参量 A，则返回一复数，其分量的实数与虚数部分分别取原复数的、朝负无穷大方向的整数部分。

【功能介绍】 对数值向负无穷大方向取整。

【实例 3.22】 将向量 A = [−1.9, −0.2, 3.1415926, 5.6, 7.0, 2.4+3.6i] 向负的方向取整。

```
>>A = [-1.9, -0.2, 3.1415926, 5.6, 7.0, 2.4+3.6i];
>>F = floor(A)
```

计算结果为：

```
F =
    Columns 1 through 4
     -2.0000          -1.0000          3.0000          5.0000
    Columns 5 through 6
     7.0000       2.0000 + 3.0000i
```

【实例讲解】 无论是负的小数还是正的小数，都要向负的并且离这个小数最近的方向逼近。

3.1.23 rem 函数——求余数

【语法说明】 R= rem(X,Y) ：返回结果 X − fix(X./Y).*Y，其中 X、Y 应为正数。若 X、Y 为浮点数，由于计算机对浮点数的表示不精确，则结果可能是不可意料的。fix(X./Y)为商数 X./Y 朝零方向取的整数部分。若 X 与 Y 为同符号的，则 rem(X,Y)返回的结果与 mod(X,Y)相同，不然，若 X 为正数，则 rem(−X,Y) = mod(−X,Y) − Y。该函数返回的结果在区间[0，sign(X)*abs(Y)]，若 Y 中有零分量，则相应地返回 NaN。

【功能介绍】 对数据求做除法后的余数。

【实例 3.23】 求向量 X=[29,48,78,348,127,90,375,438]与向量 Y=[3,4,5,6,7,8,9,2]相除后得到的余数向量。

```
>> X=[29,48,78,348,127,90,375,438];
>> Y=[3,4,5,6,7,8,9,2];
>> R = rem(X,Y)
R =
    2    0    3    0    1    2    6    0
```

【实例讲解】 用户可以直接使用数组相除的方法，计算出上面的结果。

3.1.24　ceil 函数——朝正无穷大方向取整

【语法说明】 B =ceil(A) ：对 A 的每一个元素朝正无穷大的方向取整数部分，返回与 A 同维的数组。对于复数参量 A，则返回一复数，其分量的实数与虚数部分分别取原复数的、朝正无穷大方向的整数部分。

【功能介绍】 朝正无穷大方向取整。

【实例 3.24】 对向量 A = [−1.9, −0.2, 3.1415926, 5.6, 7.0, 2.4+3.6i] 做取整运算。

```
>>A = [-1.9, -0.2, 3.1415926, 5.6, 7.0, 2.4+3.6i];
>>B = ceil(A)
```

计算结果为：

```
B =
   Columns 1 through 4
   -1.0000         0         4.0000         6.0000
   Columns 5 through 6
    7.0000    3.0000 + 4.0000i
```

3.1.25　real 函数——复数的实数部分

【语法说明】 Y = real(Z)：返回输入参量 Z 的每一个分量的实数部分。

【功能介绍】 求复数的实数部分。

【实例 3.25】 求复数 5+8i 的实数部分。

```
>> a=5+8i
a =
   5.0000 + 8.0000i
>> real(a)
ans =
     5
```

【实例讲解】 这个例子中复数的实数部分为 5。

3.1.26 imag 函数——复数的虚数部分

【语法说明】 Y = imag(Z)：返回输入参量 Z 的每一个分量的虚数部分。

【功能介绍】 求复数的虚数部分。

【实例 3.26】 求复数 5+8i 的虚数部分。

```
>> a=5+8i
a =
   5.0000 + 8.0000i
>> imag (a)
ans =
     8
```

【实例讲解】 得到的结果就为虚数部分的系数 8。

3.1.27 angle 函数——求复数的相角

【语法说明】 P = angle(Z)：返回输入参量 Z 的每一复数元素的、单位为弧度的相角，其值在区间[−π，π]上。angle(z) = imag (log(z)) = atan2 (imag(z)，real(z))。

【功能介绍】 求复数的相位角。

【实例 3.27】 求向量 ***X*** 的相位角。

```
>>X=[1-i, 2+i, 3-i, 4+i;
>>1+2i,2-2i,3+2i,4-2i;
>>1-3i,2+3i,3-3i,4+3i;
>>1+4i,2-4i,3+4i,4-4i];
>>P = angle(Z)
```

计算结果为：

```
P =
   -0.7854    0.4636   -0.3218    0.2450
    1.1071   -0.7854    0.5880   -0.4636
   -1.2490    0.9828   -0.7854    0.6435
    1.3258   -1.1071    0.9273   -0.7854
```

【实例讲解】 得到的相位角均为以弧度制表示的形式，如果变为角度值还要进行相应的转换。

3.1.28　conj 函数——复数的共轭值

【语法说明】 ZC = conj(Z)：返回参量 Z 的每一个分量的共轭复数，conj(Z) = real(Z) − i*imag(Z)。

【功能介绍】 求复数的共轭值。

【实例 3.28】 利用 conj 函数求取复数 X=5+8*i* 的共轭。

```
>> X=5+8i
X =
   5.0000 + 8.0000i
>> conj(a)
ans =
   5.0000 - 8.0000i
```

【实例 3.29】 对复数 x=1−2*i* 的综合应用演示。

```
>> x=1-2*i;
>> real(x) % 列出实部
ans =
1
>> imag(x) % 列出虚部
ans =
-2
>> conj(x)   % 计算共轭复数
ans =
1.0000 + 2.0000i
>> abs(x)   % 计算复数的大小
ans =
2.2361
>> angle(x) % 计算复数向量的夹角（以径度表示）
ans =
```

```
-1.1071
>> a=1; b=4; c=13;
>> x1=(-b+sqrt(b^2-4*a*c))/(2*a) % 以解二次方程式根的公式计算复数根
x1 =
-2.0000 + 3.0000i
>> x2=(-b-sqrt(b^2-4*a*c))/(2*a)
x2 =
-2.0000 - 3.0000i
>> y=exp(i) % 以复数指数方式表示一个复数
y =
0.5403 + 0.8415i
>> y=exp(i*pi*0.75)
y =
-0.7071 + 0.7071i
```

【实例 3.30】 利用 polar 指令绘制复数的极坐标图。

```
>> t=0:0.01:2*pi;
>> r=sin(2*t).*cos(2*t);
>> polar(t,r)
>> title('Polar plot of sin(2t)cos(2t)')
>> angle=0:2*pi/100:2*pi;
>> r=angle/(2*pi);
>> polar(angle,r)
>> title('Polar plot')
>> grid
```

上面函数得到的两个图形如图 3.26 和图 3.27 所示。

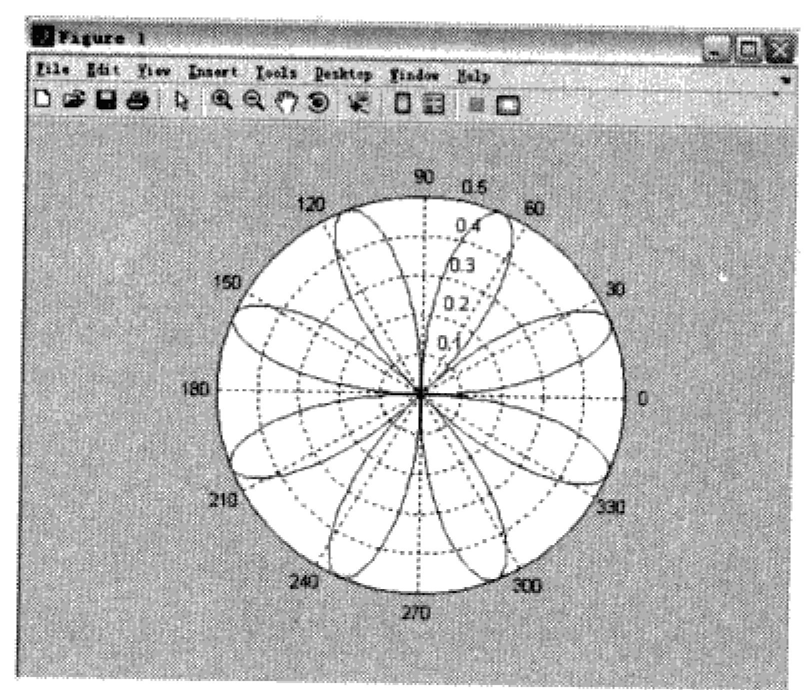


图 3.26 sin(2x)cos(2x)图形

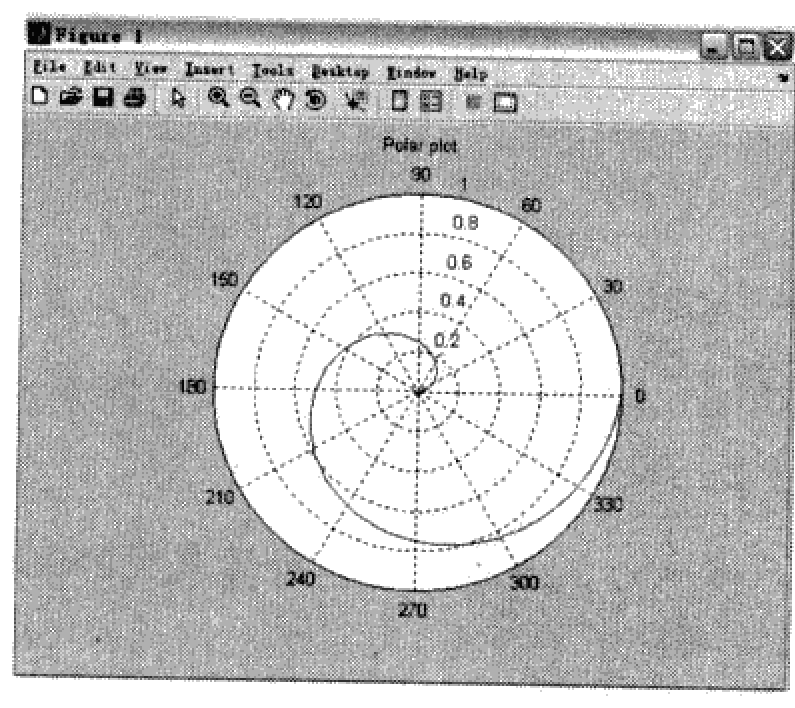


图 3.27 螺旋线图形绘制

【实例讲解】 在上面的图形中，用户使用了 ploar 函数绘制图形。关于这个函数的具体用法，请查看后面章节的介绍。

3.1.29 complex 函数——创建复数

【语法说明】

- c = complex(a,b)：用两个实数 a，b 创建复数 c=a+bi。输出参量 c 与 a、b 同型（同为向量、矩阵或多维阵列）。该函数比下列形式的复数输入更有用：a + i×b 或 a + j×b 因为 i 和 j 可能被用做其他的变量（不等于 sqrt(−1)），或者 a 和 b 不是双精度的。
- c = complex(a)：输入参量 a 作为输出复数 c 的实部。

【功能介绍】 用实数与虚数部分创建复数。

【实例 3.31】 用实部与虚部的表示方法创建复数。

```
>>a = uint8([1;2;3;4]);
>>b = uint8([4;3;2;1]);
>>c = complex(a,b)
```

计算结果为：

```
s =
   1.0000 + 4.0000i
   2.0000 + 3.0000i
   3.0000 + 2.0000i
   4.0000 + 1.0000i
```

【实例讲解】 在 MATLAB 中，提供了多种复数和实数的转换函数。complex 就是其中有重要应用的函数之一。

3.1.30 mod 函数——求模数

【语法说明】 M = mod(X,Y)：输入参量 X、Y 应为整数，此时返回余数 X −Y.*floor(X./Y)，且运算数 x 与 y 有相同的符号，则 mod(X,Y)等于 rem(X,Y)。总之，对于整数 x,y，有：mod(−x,y) = rem(−x,y)+y。若输入为实数或复数，由于浮点数在计算机上的不精确表示，该操作将导致不可预测的结果。

【功能介绍】 求模数（带符号的除法余数）。

【实例 3.32】 分别求取下面几个参量的模数。

```
>>M1 = mod(13,5)
>>M2 = mod([1:5],3)
>>M3 = mod(magic(3),3)
```

计算结果为：

```
M1 =
    3
M2 =
    1    2    0    1    2
M3 =
    2    1    0
    0    2    1
    1    0    2
```

【实例讲解】 从上面的例子中可以看出，mod 函数可以很便利地操作各种维度的矩阵。

3.1.31 nchoosek 函数——二项式系数或所有的组合数

【语法说明】

- C = nchoosek(n,k)：参量 n,k 为非负整数，返回 n! / ((n−k)! k!)，即一次从 n 个物体中取出 k 个的组合数。
- C = nchoosek(v,k)：参量 v 为 n 维向量，返回一矩阵，其行向量的分量为一次性从 v 个物体中取 k 个物体的组合数。矩阵 C 包含 C_k^n=n! / ((n−k)! k!)行与 k 列。

【功能介绍】 求组合数。

【实例 3.33】 求从 9 个编号的小球中取出 6 个有几种方法。

```
>>  C = nchoosek(9,6)
```

计算结果为：

```
     C =
84
```

【实例讲解】 容易得出，共有 84 种方法。

3.1.32 rand 函数——生成均匀分布矩阵

【语法说明】

- Y = rand(n)：返回 n*n 阶的方阵 Y，其元素均匀分布于区间(0,1)。若 n 不是一标量，在显示一出错信息。
- Y = rand(m,n)、Y = rand([m n])：返回阶数为 m*n 的，元素均匀分布于区间(0,1)上矩阵 Y。
- Y = rand(m,n,p,…)、Y = rand([m n p…])：生成阶数 m*n*p*…的，元素服从均匀分布的多维随机阵列 Y。
- Y = rand(size(A))：生成一与阵列 A 同型的随机均匀阵列 Y
- rand：该函数在每次单独使用时，都返回一随机数（服从均匀分布）。
- s = rand('state')：返回一有 35 元素的列向量 s，其中包含均匀分布生成器的当前状态。

【功能介绍】 生成均匀分布于(0,1)上的数值与阵列。

【实例 3.34】 建立一个 5×6 维的均匀分布于（0，1）上的矩阵，同时建立一个 5×5 维的均匀分布在(10，50)上的矩阵。

```
>> R1 = rand(5,6)
R1 =
    0.9501    0.7621    0.6154    0.4057    0.0579    0.2028
    0.2311    0.4565    0.7919    0.9355    0.3529    0.1987
    0.6068    0.0185    0.9218    0.9169    0.8132    0.6038
    0.4860    0.8214    0.7382    0.4103    0.0099    0.2722
    0.8913    0.4447    0.1763    0.8936    0.1389    0.1988
>> a = 10; b = 50;
>> R2 = a + (b-a) * rand(5)
R2 =
   10.6110   26.7460   43.5247   30.1125   17.7372
   39.8714   43.8489   10.7856   38.3789   37.2889
   27.8039   31.0061   37.2511   27.1557   22.1106
   47.2726   18.1059   25.1792   22.1847   31.6670
   28.6398   36.8855   43.2718   17.5861   16.0349
```

【实例讲解】 读者可以对比上面两种使用方法的差别。

【实例 3.35】 生成随机数向量并作图显示。

```
>> rand(1,9) % 产生随机数

ans =

  Columns 1 through 6

    0.9501    0.2311    0.6068    0.4860    0.8913    0.7621

  Columns 7 through 9

    0.4565    560.0185    0.8214
>> hist(ans) % 用条形图表示
>> plot(ans) % 用线性图表示
```

得到的结果如图 3.28 和图 3.29 所示。

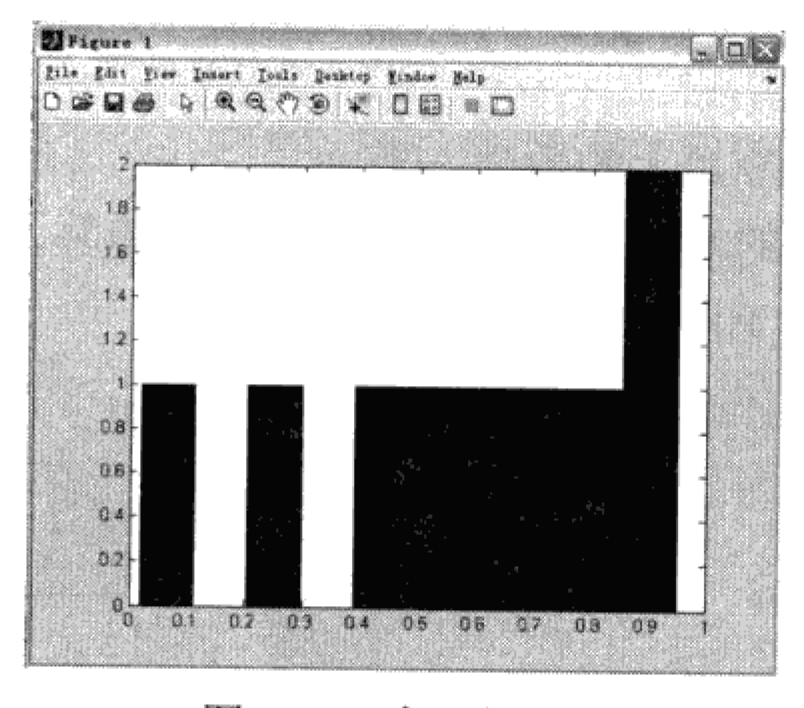


图 3.28　条形图表示

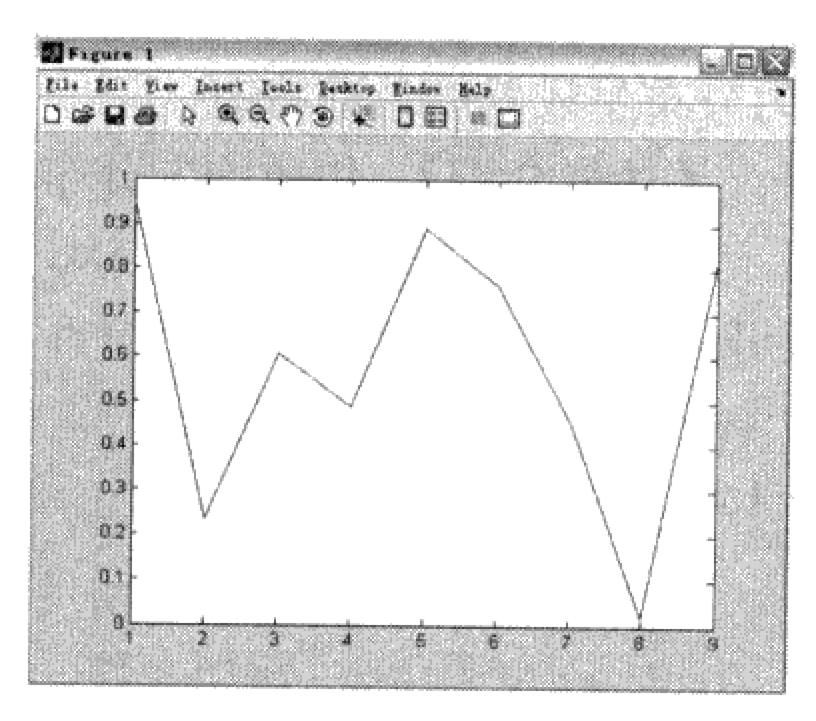


图 3.29　线性图表示

【实例讲解】 通过两种图形的表示，能够知道两种图形之间的区别，线性图表示就是准确地表示出每个点的值，通过线性图可以查出共有 9 个拐点，也就是说有 9 个不同的数值，而条形图则不能具体刻画细节，它只说明在不同的时间区域数值的聚集程度，从上面的 9 个常数中可见，没有任何数值落在[0.1 0.2]和[0.3 0.4]之间，因此在条形中体现的是这两个区域为空白。

3.1.33 randn 函数——生成服从正态分布矩阵

【语法说明】

- Y = randn(n) ：返回 n×n 阶的方阵 Y，其元素服从正态分布 N(0,1)。若 n 不是一个标量，则显示出错信息。
- Y = randn(m,n)、Y = randn([m n]) ：返回阶数为 m×n 的，元素均匀分布于区间(0,1)上矩阵 Y。
- Y = randn(m,n,p,⋯)、Y = randn([m n p⋯]) ：生成阶数 m×n×p×⋯⋯的，元素服从正态分布的多维随机阵列 Y。
- Y = randn(size(A))：生成一与阵列 A 同型的随机正态阵列 Y。
- randn：该函数在每次单独使用时，都返回一随机数（服从正态分布）。
- s = randn('state') ：返回一个有两个元素的向量 s，其中包含正态分布生成器的当前状态。

【功能介绍】 生成服从正态分布（N(0,1)）的阵列。

【实例 3.36】 生成 4×5 的服从正态分布的矩阵。

```
>>R1 = rand(4,5)
>>R2 = 0.6 + sqrt(0.1) * randn(5)
```

计算结果为：

```
R1 =
    0.2778    0.2681    0.5552    0.5167    0.8821
    0.2745    0.3710    0.1916    0.3385    0.5823
    0.9124    0.5129    0.4164    0.2993    0.0550
    0.4125    0.2697    0.1508    0.9370    0.5878
R2 =
    0.4632    0.9766    0.5410    0.6360    0.6931
    0.0733    0.9760    0.8295    0.9373    0.1775
    0.6396    0.5881    0.4140    0.6187    0.8259
    0.6910    0.7035    1.2904    0.5698    1.1134
    0.2375    0.6552    0.5569    0.3368    0.3812
```

【实例讲解】 读者可以多次在 MATLAB 中运行 randn 函数，查看每次运行的结果。

3.2 插值、拟合与查表

在大量的应用领域中，人们经常面临用一个解析函数描述数据（通常是测量值）的任务。插值与数据拟合是非常实用的数值方法，是函数逼近的重要手段。在工程生产和科学实验中，自变量 x 与因变量 y 的函数关系式有时不能直接写出，而只能得到函数在若干个点的函数值或导数值。当要求知道观测点之外的函数值时，需要估计函数值在该点的值。因此，通过已有数据信息构造出函数表达式显得尤为重要。

3.2.1 interp1 函数——一维数据插值函数

【语法说明】

- yi = interp1(x,Y,xi)：返回插值向量 yi，每一元素对应于参量 xi，同时由向量 x 与 Y 的内插值决定。参量 x 指定数据 Y 的点。若 Y 为一矩阵，则按 Y 的每列计算。yi 是阶数为 length(xi)*size(Y,2) 的输出矩阵。
- yi = interp1(Y,xi)：假定 x=1:N，其中 N 为向量 Y 的长度，或者为矩阵 Y 的行数。
- yi = interp1(x,Y,xi,method)：用指定的算法计算插值。nearest 为最近邻点插值，直接完成计算；linear 为线性插值（默认方式），直接完成计算；spline 为三次样条函数插值。
- yi = interp1(x,Y,xi,method,'extrap')：对于超出 x 范围的 xi 中的分量将执行特殊的外插值法 extrap。
- yi = interp1(x,Y,xi,method,extrapval)：确定超出 x 范围的 xi 中的分量的外插值 extrapval，其值通常取 NaN 或 0。

【功能介绍】 一维数据插值。该函数对数据点之间计算内插值，它找出一元函数 $f(x)$在中间点的数值，其中函数表达式由所给数据决定。

【实例 3.37】 根据离散点绘制出插值函数图形。

```
>> x = 0:20;
>> y = x.*sin(x);
>> x1 = 0:.25:10;
>>  x1 = 0:.20:20;
>> y1 = interp1(x,y,x1);
>> plot(x,y,'kd',x1,y1)
```

得到的结果如图 3.30 所示。

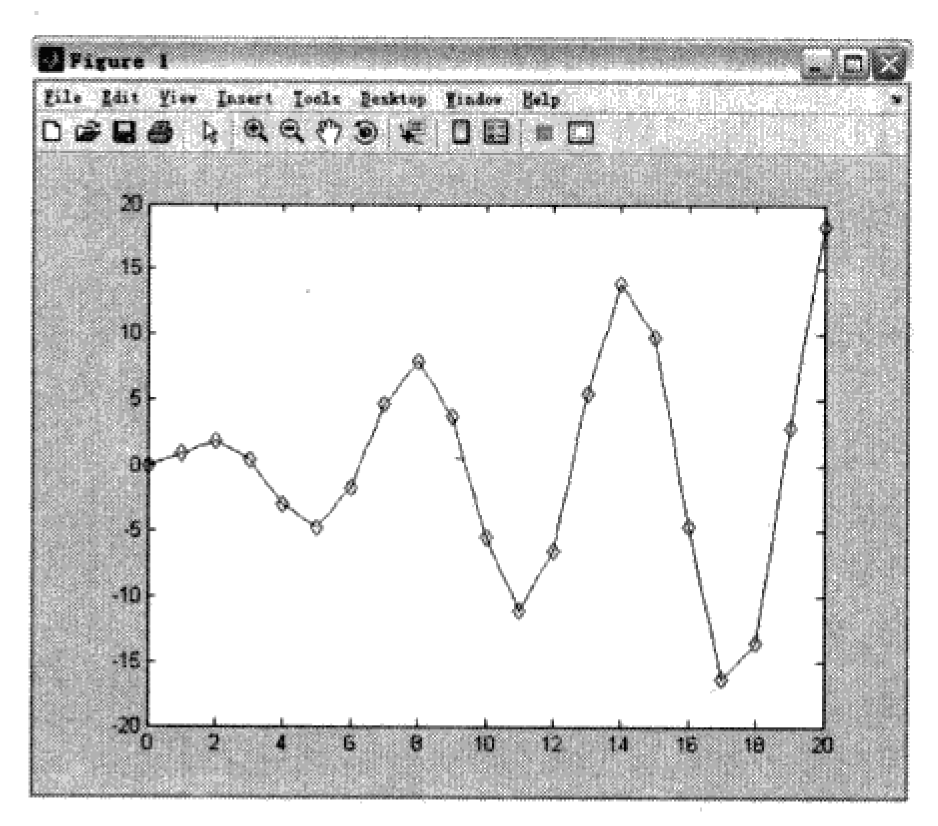


图 3.30 一元函数插值图形

【实例讲解】 上面的图形形象地显示了一元函数插值的过程和结果。

3.2.2 interp2 函数——二维数据内插值

【语法说明】

- ZI = interp2(X,Y,Z,XI,YI) ：返回矩阵 ZI，其元素包含对应于参量 XI 与 YI（可以是向量、或同型矩阵）的元素。用户可以输入行向量和列向量 Xi 与 Yi，此时，输出向量 Zi 与矩阵 meshgrid(xi,yi)是同型的。同时取决于由输入矩阵 X、Y 与 Z 确定的二维函数 Z=f(X,Y)。
- ZI = interp2(Z,XI,YI) ：默认地，X=1:n、Y=1:m，其中

[m,n]=size(Z)。再按第一种情形进行计算。

■ ZI = interp2(Z,n) ：作 n 次递归计算，在 Z 的每两个元素之间插入它们的二维插值，这样，Z 的阶数将不断增加。interp2(Z)等价于 interp2(z,1)。

■ ZI = interp2(X,Y,Z,XI,YI,method) ：用指定的算法 method 计算二维插值。linear 为双线性插值算法（默认算法），nearest 为最临近插值，spline 为三次样条插值，cubic 为双三次插值。

【功能介绍】 完成二维的数据插值。

【实例 3.38】 查看二维数据插值并绘图。

```
>>[X,Y] = meshgrid(-3:.25:3);       %定义二维数据的范围和步长,生成同型阵列
>>Z = peaks(X,Y);                   %绘制等高线
>>[XI,YI] = meshgrid(-3:.125:3);    %定义二维数据的范围和步长,生成同型阵列
>>Z1 = interp2(X,Y,Z,XI,YI);
>>surfl(X,Y,Z);hold on;             %绘制二维图形
>>surfl(XI,YI,Z1+15)
>>axis([-3 3 -3 3 -5 20]);shading flat
>>hold off
```

得到的结果如图 3.31 所示。

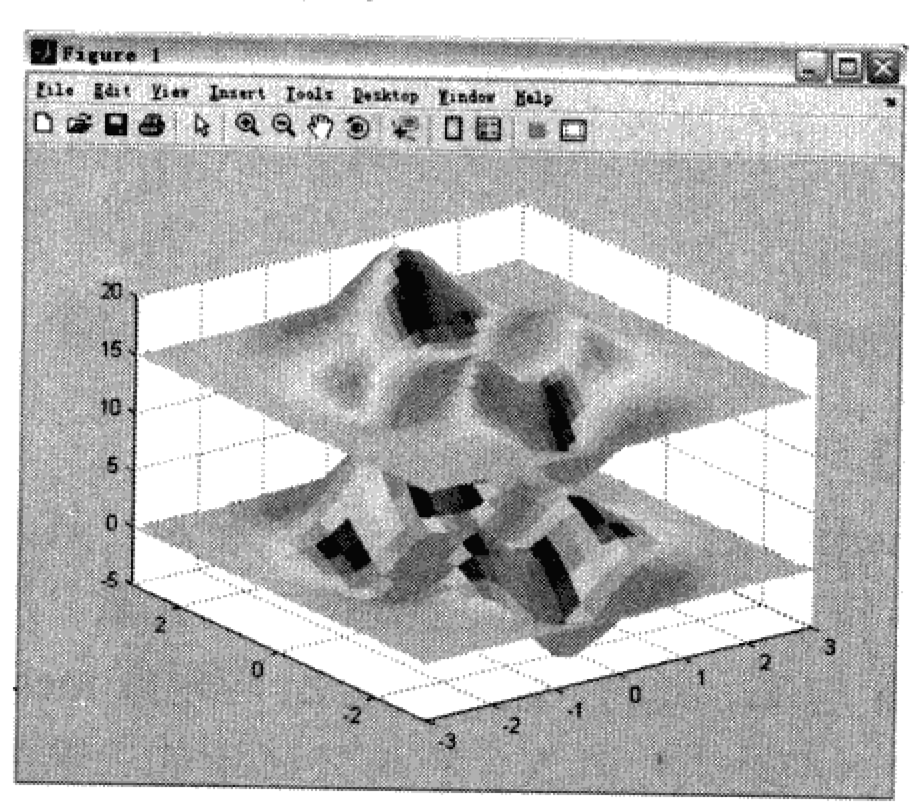


图 3.31　二维插值图形

【实例讲解】 在上面的截图中，图形区域颜色显示的是不同的数值。

3.2.3　interp3 函数——三维数据插值

【语法说明】

- VI = interp3(X,Y,Z,V,XI,YI,ZI) ：求出由参量 X,Y,Z 决定的三元函数 V=V(X,Y,Z)在点（XI,YI,ZI）的值。参量 XI,YI,ZI 是同型阵列或向量。若向量参量 XI,YI,ZI 是不同长度、不同方向（行或列）的向量，这时输出参量 VI 与 Y1,Y2,Y3 为同型矩阵。Y1,Y2,Y3 为用函数 meshgrid(XI,YI,ZI)生成的同型阵列。若插值点(XI,YI,ZI)中有位于点(X,Y,Z)之外的点，则相应地返回特殊变量值 NaN。
- VI = interp3(V,XI,YI,ZI) ：默认地，X=1:N，Y=1:M，Z=1:P，其中，[M,N,P]=size(V)，再按上面的情形计算。
- VI = interp3(V,n)：作 n 次递归计算，在 V 的每两个元素之间插入它们的三维插值。这样，V 的阶数将不断增加。interp3(V)等价于 interp3(V,1)。
- VI = interp3(…,method)：用指定的算法 method 作插值计算。linear 为线性插值（默认算法），cubic 为三次插值，spline 为三次样条插值，nearest 为最邻近插值。

【功能介绍】 完成三维数据插值。

【实例 3.39】 画出三维数据插值结果图。

```
>>[x,y,z,v] = flow(20);
>>[xx,yy,zz]  =   meshgrid(.1:.25:10,   -4:0.2:4,
-4:.2:4);
>>vv = interp3(x,y,z,v,xx,yy,zz);
>>slice(xx,yy,zz,vv,[6 9.5],[1 2],[ -2 .2]);
```

得到的结果如图 3.32 所示。

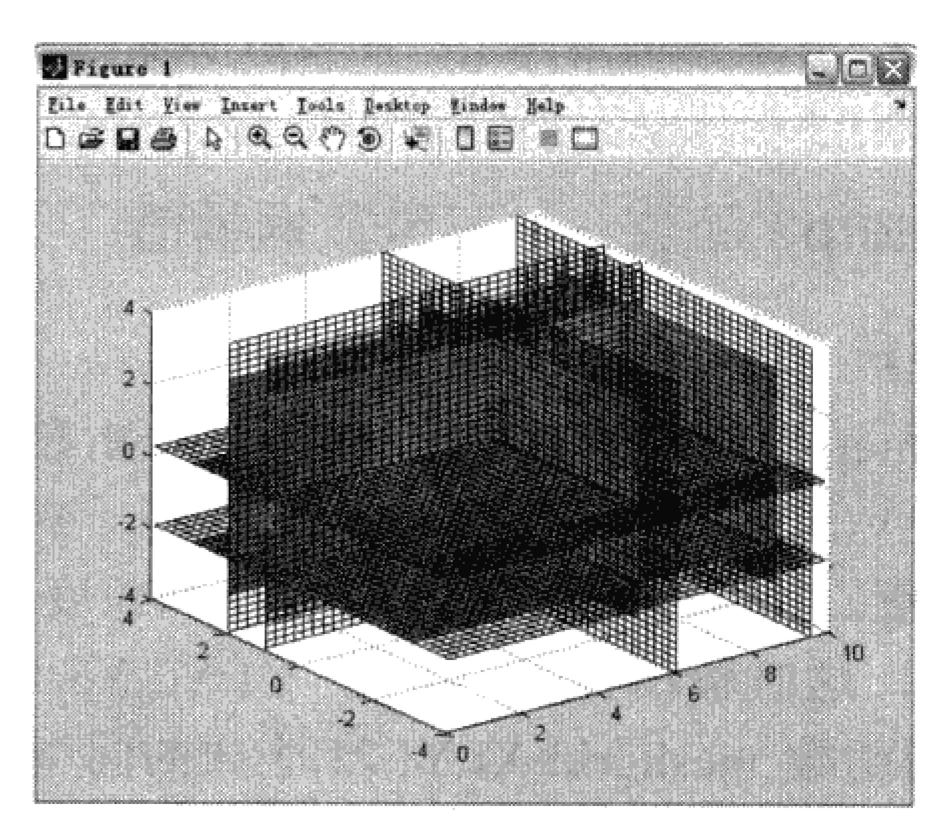


图 3.32　三维插值图

3.2.4　interpn 函数——n 维数据插值

【语法说明】

- VI = interpn(X1,X2,,⋯,Xn,V,Y1,Y2,⋯,Yn)：返回由参量 X1,X2,…,Xn,V 确定的 n 元函数 V=V(X1,X2,…,Xn) 在点（Y1,Y2,…,Yn）处的插值。参量 Y1,Y2,…,Yn 是同型的矩阵或向量。若 Y1,Y2,…,Yn 是向量，则可以是不同长度，不同方向（行或列）的向量。
- VI = interpn(V,Y1,Y2,⋯,Yn)：默认地，X1=1:size(V,1)，X2=1:size(V,2)，…，Xn=1:size(V,n)，再按上面的情形计算。
- VI = interpn(V,ntimes)：作 ntimes 次递归计算，在 V 的每两个元素之间插入它们的 n 维插值。这样，V 的阶数将不断增加。interpn(V)等价于 interpn(V,1)。

【功能介绍】 完成 *n* 维数据插值。

3.2.5　spline 函数——三次样条插值

【语法说明】 yi=spline（x,y,xi）：x、y 为插值点的向量，xi 为

所求点的横坐标值，yi 为求得的纵坐标值。

【功能介绍】 通过三次样条插值求函数值。

【实例 3.40】 绘制由离散点插值得到的函数图形。

```
>>  x=0 : 16

x =

  Columns 1 through 14

    0   1   2   3   4   5   6   7   8   9   10   11   12   13

  Columns 15 through 17

   14   15   16

>>  y=tan(pi*x/20);
>> xi=linspace(0, 16);
>> yi=spline(x, y, xi);
>>  plot(x, y, 'o', xi, yi);title('Spline fit')
```

得到的结果如图 3.33 所示。

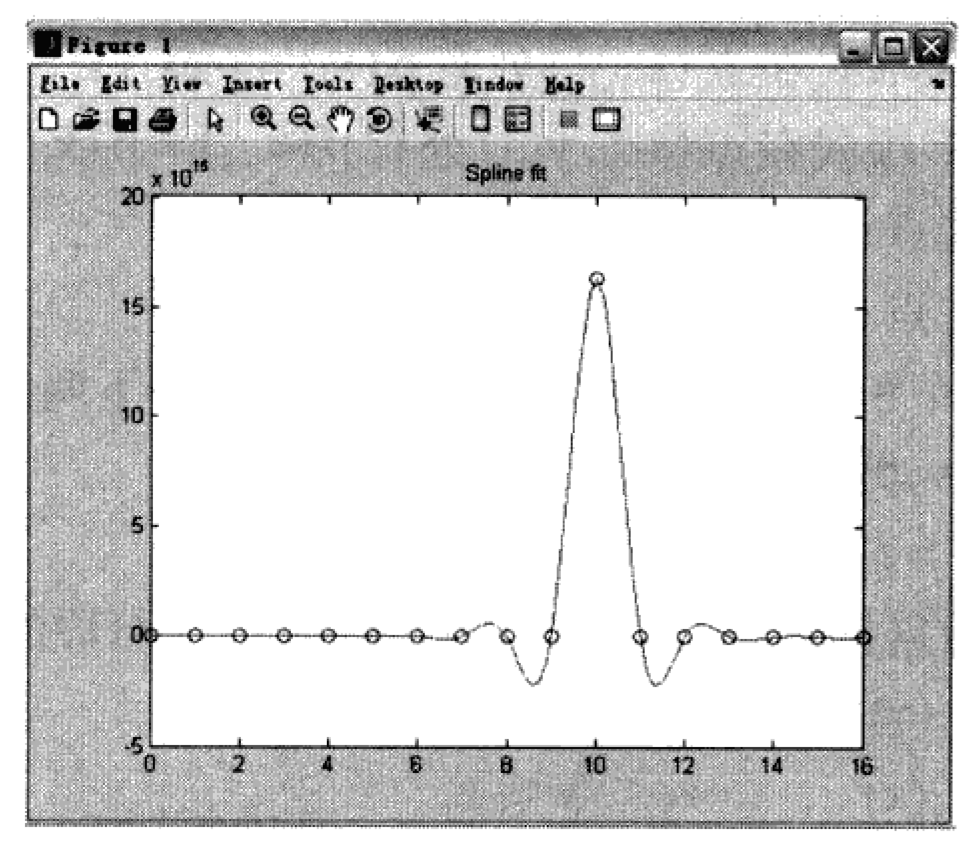


图 3.33　数据插值

【实例讲解】 x 为 0～16 的离散数值，y=tan(pi*x/20)，xi 为 linspace 函数，yi 为基于 xi 的纵坐标插值。

3.2.6 interpft 函数——用快速 Fourier 算法作一维插值

【语法说明】

- y = interpft(x,n) ：返回包含周期函数 x 在重采样的 n 个等距的点的插值 y。若 length(x)=m，且 x 有采样间隔 dx，则新的 y 的采样间隔 dy=dx*m/n。注意的是必须 n≥m。若 x 为一矩阵，则按 x 的列进行计算。返回的矩阵 y 有与 x 相同的列数，但有 n 行。
- y = interpft(x,n,dim) ：沿着指定的方向 dim 进行计算。

【功能介绍】 用快速 Fourier 算法作一维插值。

3.2.7 spline 函数——三次样条数据插值

【背景知识】 由于在上面讲解的几种插值中，过两点 (x_i,y_i) 和 (x_{i+1},y_{i+1}) 只能确定一条直线，而通过一点的三次多项式曲线有无穷多条。为了使通过中间断点的三次多项式曲线具有唯一性，要增加如下限制性条件：

- 三次多项式在点 (x_i,y_i) 处有 $p_i'(x_i)=p_i''(x_i)$;
- 三次多项式在点 (x_{i+1},y_{i+1}) 处有 $p_i'(x_{i+1})=p_i''(x_{i+1})$;
- p(x)在点 (x_i,y_i) 处的斜率是连续的；
- p(x)在点 (x_i,y_i) 处的曲率是连续的。

对于第一个和最后一个多项式，人为规定如下两个条件：

- $p_1'''(x)=p_2'''(x)$
- $p_n'''(x)=p_{n-1}'''(x)$

因此，对数据拟合的三次样条函数 p(x)是一个分段的三次多项式：

$$p(x)=\begin{cases} p_1(x) & x_1 \leqslant x \leqslant x_2 \\ p_2(x) & x_2 \leqslant x \leqslant x_3 \\ \cdots & \cdots \\ p_n(x) & x_n \leqslant x \leqslant x_{n+1} \end{cases}$$

，其中每段 $p_i(x)$ 都是三次多项式。

【语法说明】

- yy = spline(x,y,xx)：对于给定的离散的测量数据 x、y（称为断点），要寻找一个三项多项式使其逼近每对数据(x,y)点的曲线。
- pp = spline(x,y)：返回由向量 x 与 y 确定的分段样条多项式的系数矩阵 pp，它可用于函数 ppval、unmkpp 的计算。

【功能介绍】 这个函数用三次样条插值计算出由向量 x 与 y 确定的一元函数 $y=f(x)$在其他点处的值。若参量 y 是一矩阵，则以 y 的每一列和 x 配对，再分别计算由它们确定的函数在点 xx 处的值。

【实例 3.41】 对离散地分布在 $y=\mathrm{e}^x \sin x$ 函数曲线上的数据点进行样条插值计算。

```
>>x = [0 1 3 5 9 11 12.5 13.9 19.12 20];
>>y = exp(x).*sin(x);
   >>xx = 0:.20:20;
   >>yy = spline(x,y,xx);
>>plot(x,y,'o',xx,yy)
```

得到的结果如图 3.34 所示。

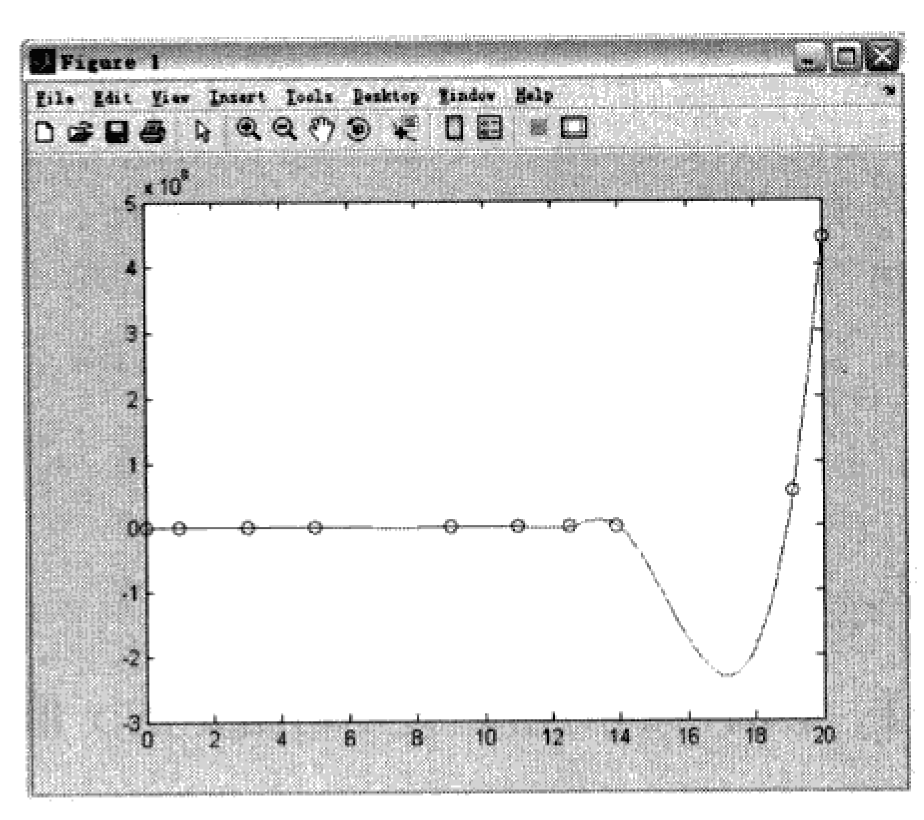


图 3.34　三次样条插值

【实例讲解】 读者可以将上面的三次样条插值的图形和二次样条插值结果进行必要的比较。

3.2.8 table1 函数—— 一维查表函数

【语法说明】 Y = table1(TAB,X0) ：返回用表格矩阵 TAB 中的行线性插值元素，对 X0（TAB 的第一列查找 X0）进行线性插值得到的结果 Y。矩阵 TAB 是第一列包含关键值，而其他列包含数据的矩阵。X0 中的每一元素将相应地返回一线性插值行向量。矩阵 TAB 的第一列必须是单调的。

【功能介绍】 一维查表函数。

【实例 3.42】 根据给定的 x 值通过查表的方式，找到线形插值的结果 y 元素。

```
>>tab = [(1:4)' hilb(4)]
>>y = table1(tab,[1 2.3 3.6 4])
```

查表结果为：

```
tab =
      1.0000    1.0000    0.5000    0.3333    0.2500
      2.0000    0.5000    0.3333    0.2500    0.2000
      3.0000    0.3333    0.2500    0.2000    0.1667
      4.0000    0.2500    0.2000    0.1667    0.1429
Warning: TABLE1 is obsolete and will be removed in future
versions.  Use INTERP1 or INTERP1Q instead.
> In D:\MATLABR12\toolbox\matlab\polyfun\table1.m at
line 31
y =
    1.0000    0.5000    0.3333    0.2500
    0.4500    0.3083    0.2350    0.1900
    0.2833    0.2200    0.1800    0.1524
    0.2500    0.2000    0.1667    0.1429
```

【实例讲解】 上例得到的结果就是对给定的 x 值查表得到的线性插值的结果。

3.2.9 table2 函数——二维查表

【语法说明】 Z = table1(TAB,X0,Y0) ：返回用表格矩阵 TAB

中的行与列交叉线性插值元素，对 X0（TAB 的第一列查找 X0）进行线性插值，对 Y0（TAB 的第一行查找 Y0）进行线性插值，对上述两个数值进行交叉线性插值，得到的结果为 Z。矩阵 TAB 是第一列与第一行都包含关键值，而其他的元素包含数据的矩阵。TAB(1,1)的关键值将被忽略。[X0,Y0]中的每点将相应地返回一线性插值。矩阵 TAB 的第一行与第一列必须是单调的。

【功能介绍】 二维查表操作。

【实例 3.43】 演示二维查表。

```
>>tab = [NaN 1:4; (1:4)' magic(4)]
>>y = table2(tab,[2 3 3.7],[1.3 2.3 4])
```

查表的结果为：

```
tab =
     NaN     1      2      3     4
       1    16      2      3    13
       2     5     11     10     8
       3     9      7      6    12
       4     4     14     15     1
  Warning: TABLE2 is obsolete and will be removed in
future versions.  Use INTERP2 instead.
  > In D:\MATLABR12\toolbox\matlab\polyfun\table2.m at
line 24
  Warning: TABLE1 is obsolete and will be removed in
future versions.  Use INTERP1 or INTERP1Q instead.
  > In D:\MATLABR12\toolbox\matlab\polyfun\table1.m at
line 31
  In  D:\MATLABR12\toolbox\matlab\polyfun\table2.m  at
line 29
  Warning: TABLE1 is obsolete and will be removed in
future versions.  Use INTERP1 or INTERP1Q instead.
  > In D:\MATLABR12\toolbox\matlab\polyfun\table1.m at
line 31
  In  D:\MATLABR12\toolbox\matlab\polyfun\table2.m  at
line 31
  y =
     6.8000   10.7000    8.0000
     8.4000    6.7000   12.0000
     7.4200   12.0200    4.3000
```

【实例讲解】 上例得到的结果就是对给定的 x 值查表得到的交叉插值的结果。

3.3 数据分析函数

将工程及科学实验所量测的数据做分析，是实验评估一项的极重要的工作。这样的分析工作可以从简单的运算例如计算平均值，到复杂的矩阵运算例如计算标准差(deviation)。这些量测可称为统计量测，因为测量这些数据含有统计性质。比方说我们量测每日的相对湿度，它的变化是和气温高低、晴天或是下雨、 地形、纬度等息息相关的，这些因素都会不时地发生变化。就像我们可以从统计资料中计算其特性，我们也能够利用电脑依照预设的统计特性来产生特定的数据。在这节中我们将介绍一些 MATLAB 基本分析数据的函数。

3.3.1 max 函数——最大值函数

【语法说明】

- max(x)：找出 x 向量的最大值。
- max(x,y)：找出 x 及 y 阵列的最大值，会有两个极值分属 x 及 y 阵列。
- [y,i]=max(x)：找出 x 阵列的最大值以 y 显示，其在 x 阵列的位置以 i 显示。

【功能介绍】 求数据矩阵或向量中的最大值。

【实例 3.44】 求向量 *x*=[1 2 3 4 5 6 8 9 4 5 2 3 7 4 3 9]中元素的最大值，并将其与向量 *y*=[1 4 5 6 7 8 9 3 5 2 5 76 5 56 9 0]进行比较。

```
>> x=[1 2 3 4 5 6 8 9 4 5 2 3 7 4 3 9]

x =

  Columns 1 through 10
```

```
      1     2     3     4     5     6     8     9     4     5

    Columns 11 through 16

      2     3     7     4     3     9

>> max(x)

ans =

      9

>> y=[1 4 5 6 7 8 9 3 5 2 5 76 5 56 9 0]  %y与x的维数必须相同

y =

    Columns 1 through 10

      1     4     5     6     7     8     9     3     5     2

    Columns 11 through 16

      5    76     5    56     9     0

>> max(x,y)

ans =

    Columns 1 through 10

      1     4     5     6     7     8     9     9     5     5

    Columns 11 through 16

      5    76     7    56     9     9
```

【实例讲解】 如上例中第二个函数中两个向量x和y的元素个数必须相同，而且得到的结果是两个向量中对应元素的最大值组成

的阵列。

【实例 3.45】 求向量 x=[1 2 3 4 5 6 8 9 4 5 2 3 7 4 3 9]中元素的最大值，并找出其所在阵列中的位置。

```
>> [z,i]=max(x)

z =

     9


i =

     8
```

【实例讲解】 i 为最大值第一次出现的位置，以后再出现同样的数值也不做记录。

3.3.2 min 函数——求最小值函数

【语法说明】

- min(x)：找出 x 阵列的最小值。
- min(x,y)：找出 x 及 y 阵列的最小值，会有两个极值分属 x 及 y 阵列。
- [y,i]=min(x)：找出 x 阵列的最小值以 y 显示，其在 x 阵列的位置以 i 显示。

【功能介绍】 求向量中元素的最小值。

【实例 3.46】 求向量 x=[1 2 3 4 5 6 8 9 4 5 2 3 7 4 3 9]中元素的最小值，并将其与向量 y=[1 4 5 6 7 8 9 3 5 2 5 76 5 56 9 0]进行按小比较。

```
>> x=[1 2 3 4 5 6 8 9 4 5 2 3 7 4 3 9]

x =

  Columns 1 through 10

     1     2     3     4     5     6     8     9     4     5
```

```
  Columns 11 through 16

     2     3     7     4     3     9

>> y=[1 4 5 6 7 8 9 3 5 2 5 76 5 56 9 0]

y =

  Columns 1 through 10

     1     4     5     6     7     8     9     3     5     2

  Columns 11 through 16

     5    76     5    56     9     0

>> min(x)

ans =

     1

>> min(x,y)

ans =

  Columns 1 through 10

     1     2     3     4     5     6     8     3     4     2

  Columns 11 through 16

     2     3     5     4     3     0
```

【实例讲解】 如上例中第二个函数中两个向量 x 和 y 的元素个数必须相同，而且得到的结果是两个向量中对应元素的最小值组成的阵列。

【实例 3.47】 求向量 x=[1 2 3 4 5 6 8 9 4 5 2 3 7 4 3 9]中元素的最小值，并找出其所在阵列中的位置。

```
x =

  Columns 1 through 10

     1     2     3     4     5     6     8     9     4     5

  Columns 11 through 16

     2     3     7     4     3     9
>> [z,i]=min(x)

z =

     1


i =

     1
```

【实例讲解】 *i* 为最小值第一次出现的位置，以后再出现同样的数值也不做记录。

3.3.3 mean 函数——平均值计算

【语法说明】 mean(x)：找出 x 向量的平均值。

【功能介绍】 求向量的平均值。

【实例 3.48】 求向量 *x*=[1 2 3 4 5 6 8 9 4 5 2 3 7 4 3 9]中元素的平均值。

```
>> x=[1 2 3 4 5 6 8 9 4 5 2 3 7 4 3 9]

x =

  Columns 1 through 10

     1     2     3     4     5     6     8     9     4     5

  Columns 11 through 16
```

```
   2     3     7     4     3     9

>> mean(x)

ans =

4.6875
```

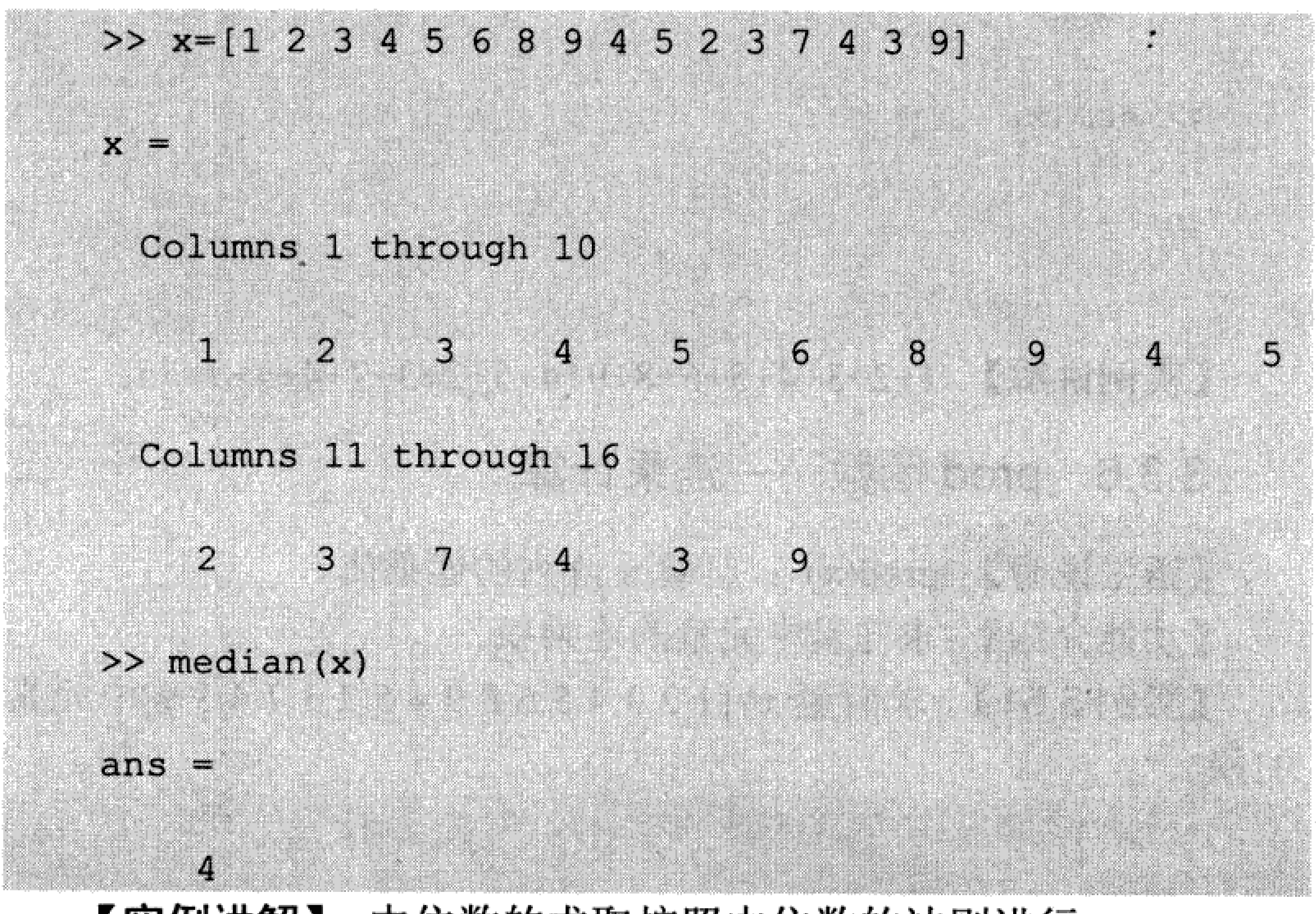
【实例讲解】 对向量求平均就是将向量中所有的元素求和，再除以这个向量中元素的总个数。

3.3.4 median 函数——中位数计算

【语法说明】 median(x)：找出 x 阵列的中位数。

【功能介绍】 求向量的中位数。

【实例 3.49】 求向量 x=[1 2 3 4 5 6 8 9 4 5 2 3 7 4 3 9]中元素的中位数。



```
>> x=[1 2 3 4 5 6 8 9 4 5 2 3 7 4 3 9]

x =

  Columns 1 through 10

   1     2     3     4     5     6     8     9     4     5

  Columns 11 through 16

   2     3     7     4     3     9

>> median(x)

ans =

   4
```

【实例讲解】 中位数的求取按照中位数的法则进行。

3.3.5 sum 函数——求和

【语法说明】 sum(x)：计算 x 向量的总和值。

【功能介绍】 求向量中元素的和。

【实例 3.50】 求向量 x=[1 2 3 4 5 6 8 9 4 5 2 3 7 4 3 9]中元素的和。

```
>> x=[1 2 3 4 5 6 8 9 4 5 2 3 7 4 3 9]

x =

  Columns 1 through 10

     1     2     3     4     5     6     8     9     4     5

  Columns 11 through 16

     2     3     7     4     3     9

>> sum(x)

ans =

    75
```

【实例讲解】 1+2+3+4+5+6+8+9+4+5+2+3+7+4+3+9=75。

3.3.6 prod 函数——连乘计算

【语法说明】 prod(x)：计算 x 阵列的连乘值。

【功能介绍】 求向量中元素的连乘积。

【实例 3.51】 求向量 x=[1 2 3 4 5 6 8 9 4 5 2 3 7 4 3 9]中元素的积。

```
>> x=[1 2 3 4 5 6 8 9 4 5 2 3 7 4 3 9]

x =
```

```
  Columns 1 through 10

     1     2     3     4     5     6     8     9     4     5

  Columns 11 through 16

     2     3     7     4     3     9

>> prod(x)

ans =

  4.7029e+009
```

【实例讲解】 1×2×3×4×5×6×8×9×4×5×2×3×7×4×3×9=4.7029e+009。

3.3.7　cumsum 函数——累积总和值

【语法说明】 cumsum(x)：计算 x 阵列的累积总和值。

【功能介绍】 计算累积总和值。

【实例 3.52】 求向量 x=[1 2 3 4 5 6 8 9 4 5 2 3 7 4 3 9]中元素的累积总和。

```
>> x=[1 2 3 4 5 6 8 9 4 5 2 3 7 4 3 9]

x =

  Columns 1 through 10

     1     2     3     4     5     6     8     9     4     5

  Columns 11 through 16

>> cumsum(x)

ans =
```

```
  Columns 1 through 10

     1     3     6    10    15    21    29    38    42    47

  Columns 11 through 16

    49    52    59    63    66    75
```

【实例讲解】 累积总和向量中的元素是这样得到的，每个位置的元素都为原向量元素的这个位置和之前所有元素的和，而最后一个元素是前面所有元素的和，所以最后一个元素的值与 sum 求到的值相同。

3.3.8 cumprod 函数——累积连乘

【语法说明】 cumprod(x) ：计算 x 阵列的累积连乘值。

【功能介绍】 求向量的累积连乘。

【实例 3.53】 求向量 *x*=[1 2 3 4 5 6 8 9 4 5 2 3 7 4 3 9]中元素的累积总积。

```
>> x=[1 2 3 4 5 6 8 9 4 5 2 3 7 4 3 9]

x =

  Columns 1 through 10

     1     2     3     4     5     6     8     9     4     5

  Columns 11 through 16

     2     3     7     4     3     9

>> cumprod(x)

ans =

  1.0e+009 *
```

```
  Columns 1 through 6

    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000

  Columns 7 through 12

    0.0000    0.0001    0.0002    0.0010    0.0021    0.0062

  Columns 13 through 16

    0.0435    0.1742    0.5225    4.7029
```

【实例讲解】 累积总和向量中的元素是这样得到的，每个位置的元素都为原向量元素的这个位置和之前所有元素的积，而最后一个元素是前面所有元素的积，所以最后一个元素的值与prod求到的值相同。

【实例3.54】 综合运用上面所讲到的所有的函数。

```
>> rains % rains 为一个 2x1 的阵列
rains =
126.8 148.5 173.0 148.4 194.7 208.9
328.8 300.7 268.3 210.5 278.4 321.5
>> avg_rain=mean(rains) % 将rains 阵列中的每一行的平均值列出
avg_rain =
227.8000 224.6000 220.6500 179.4500 236.5500 265.2000
>> avg_rain=mean(avg_rain) % 将上述阵列中的平均值列出
avg_rain =
225.7083
>> max_rain=max(rains) % 将rains 阵列中的每一行的最大值列出
max_rain =
328.8000 300.7000 268.3000 210.5000 278.4000 321.5000
>> [max_rain,x]=max(rains) % 将rains 阵列中的每一行的最大
值及其位置列出
max_rain =
328.8000 300.7000 268.3000 210.5000 278.4000 321.5000
x =
2 2 2 2 2 2
>> min_rain=min(rains) % 将rains 阵列中的每一行的最小值列出
```

```
min_rain =
126.8000 148.5000 173.0000 148.4000 194.7000 208.9000
>> s_sort=sort(rains) % 将 rains 阵列的值由小到大做排序
s_sort =
126.8000 148.5000 173.0000 148.4000 194.7000 208.9000
328.8000 300.7000 268.3000 210.5000 278.4000 321.5000
>> x=[1 2 3 4 5];
>> sum(x) % 将 x 阵列的值做总和
ans =
15
>> prod(x) % 将 x 阵列的值做连乘
ans =
120
>> cumsum(x) % 将 x 阵列的值累积后做总和
ans =
1 3 6 10 15
>> cumprod(x) % 将 x 阵列的值累积后做连乘
ans =
1 2 6 24 120
```

【实例讲解】 读者还可以将矩阵的其他操作运用在矩阵中。

3.3.9 关系及逻辑运算

在执行关系及逻辑运算时，MATLAB 将输入的不为零的数值都视为真（True），而为零的数值则视为假（False）。运算的输出值将判断为真则以 1 表示，为假则以 0 表示。MATLAB 提供以下的关系判断及逻辑的运算符。

- < ：小于。
- <= ：小于等于。
- > ：大于。
- >= ：大于等于。
- == ：等于。
- ~= ：不等于。
- & ：逻辑 and （逻辑与）。
- | ：逻辑 or （逻辑或）。

□　~　：逻辑 not （逻辑非）。

【实例 3.55】 利用上面的几个逻辑运算符对向量进行操作。

```
>> a=1:5, b=5-a,
a =
1 2 3 4 5
b =
4 3 2 1 0
>> tf1= a>4
tf =
0 0 0 0 1
>> tf2= a==b
tf =
0 0 0 0 0
>> tf3= b-(a>2)
tf =
4 3 1 0 -1
>> tf4= ~(a>4)
tf =
1 1 1 1 0
>> tf5= (a>2)&(a<6)
tf =
0 0 1 1 1
```

【实例讲解】 通过上面的例子可以看出，经过逻辑运算后的数据阵列为 1 和 0 的组合，在 $a>4$ 的判断中，只有最后一个数值 5> 4 前面的 4 个元素都不大于 4。所以，最后得到的结果是[0 0 0 0 1]；同理，对于 $a==b$ 的判断也如此，a 中的元素和 b 中的元素对不对应相等，所以得到的结果为[0 0 0 0 0]；对于～($a>4$)这类取非的操作，就是将 $a>4$ 的结果全部取反，即原来的[0 0 0 0 1]取反操作后变为[1 1 1 1 0]。

【实例 3.56】 利用关系及逻辑运算产生一个不连续的信号。

```
>> x=linspace(0,10,100); % 产生数据
>> y=sin(x); % 产生 sine 函数
>> z=(y>=0).*y; % 将 sin(x) 的负值设为零
>> z=z + 0.9*(y<0); % 再将上式的值加上 0.9
>> z=(x<6).*z; % 将大于 x=6 以后的值置零
>> hold on
```

```
>> plot(x,z)
>> xlabel('x'),ylabel('z=f(x)')
>>  hold off
```

得到的结果如图 3.35 所示。

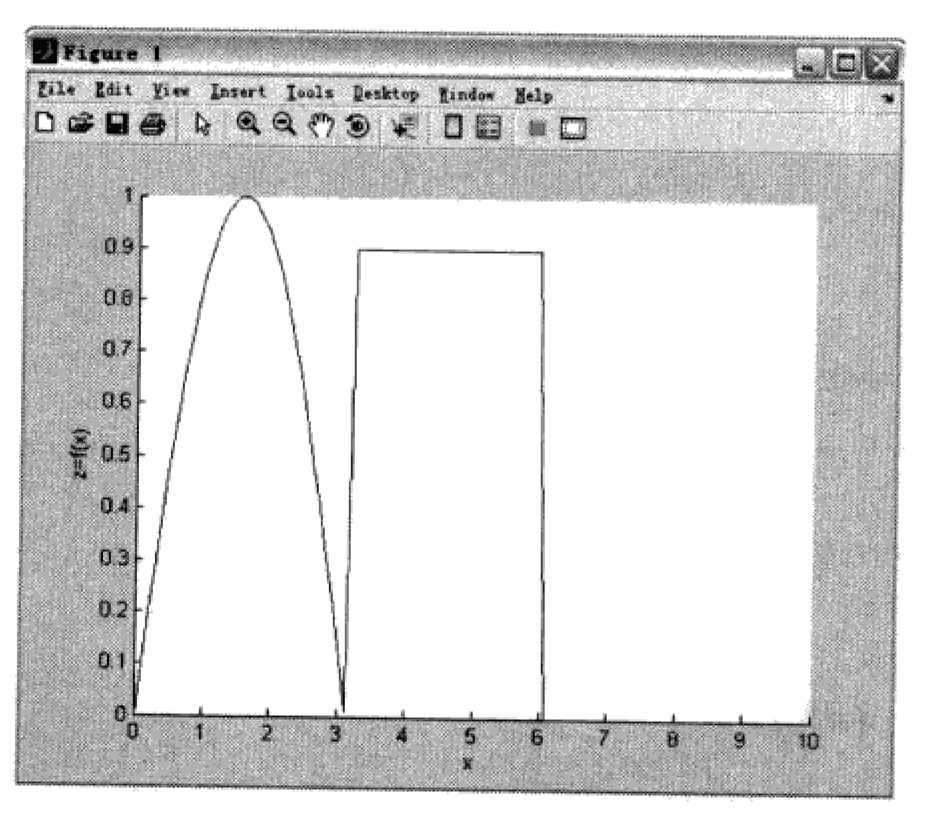


图 3.35　不连续信号

【实例讲解】 使用逻辑关系产生不连续信号是十分有效的方法。

3.4　数值微积分

3.4.1　quad 函数——一元函数的数值积分

【语法说明】

- q = quad(fun,a,b) ：近似地从 a 到 b 计算函数 fun 的数值积分，误差为 10^{-6}。若给 fun 输入向量 x，应返回向量 y，即 fun 是一单值函数。
- q = quad(fun,a,b,tol)：用指定的绝对误差 tol 代替默认误差。tol 越大，函数计算的次数越少，速度越快，但结果精度变小。
- q = quad(fun,a,b,tol,trace,p1,p2,…) ：将可选参数 p1,p2,等传

递给函数 fun(x,p1,p2,)，再作数值积分。若 tol=[]或 trace=[]，则用默认值进行计算。

- [q,n] = quad(fun,a,b,…) ：同时返回函数计算的次数 n。

【功能介绍】 采用自适应 Simpleson 积分法对数值进行定积分运算。

【实例 3.57】 对函数 $y=\dfrac{-5x^2}{x^3+3x^2-8}$ 进行数值积分。

```
>> fun = inline('-5*x.^2./(x.^3+3*x.^2-8)');
>>Y1 = quad(fun,0,2)
>>Y2 = quadl(fun,0,2)
```

计算结果为：

```
Y1 =
   -1.2554
Y2 =
    0.0281
```

【实例讲解】 通过对上例中的算式进行积分，并求出积分值。*a*、*b* 分别表示积分的上下限。

3.4.2 quad8 函数——牛顿-康兹法求积分

【语法说明】 quad8('function',a,b)，其中 function 是一已定义函数的名称（如 sin、cos、sqrt、log 等），而 *a*、*b* 是积分的下限和上限。

【功能介绍】 利用牛顿-康兹法求积分。

【实例 3.58】 求 $y=\sqrt{x}$ 在区间[0，4]上的积分。

```
>> a=0

a =

     0

>> b=4

b =

     4
```

```
>> kq=quad('sqrt',a,b)

kq =

    5.3333

>>  kq8=quad8('sqrt',a,b)
Warning: QUAD8 is obsolete.  We use QUADL instead.
> In quad8 at 32

kq8 =

5.3333
```

【实例讲解】 当使用 quad8 作为函数时，MATLAB 系统发出了警告，说 quad8 比较陈旧了，所以用 quadl 代替这个函数，但是不会影响它的计算结果。

【实例 3.59】 使用内联函数进行积分运算演示。

```
>> x=-1:0.17:2;
>> y=humps(x);
>> area1=trapz(x,y)
area =
25.9174
>> x=-1:0.07:2;
>> y=humps(x);
>> area2=trapz(x,y)
area =
26.6243
>> area3=quad('hump', -1,2)
area =
26.3450
>> area4=quad8('hump', -1,2)
area =
26.3450
```

【实例讲解】 在上面的例子中，使用不同的积分方法计算结果。读者可以对比计算的结果。

3.4.3　trapz 函数——用梯形法进行数值积分

【语法说明】

- T = trapz(Y)：用等距梯形法近似计算 Y 的积分。若 Y 是一向量，则 trapz(Y)为 Y 的积分；若 Y 是一矩阵，则 trapz(Y)为 Y 的每一列的积分；若 Y 是一多维阵列，则 trapz(Y)沿着 Y 的第一个非单元集的方向进行计算。
- T = trapz(X,Y)：用梯形法计算 Y 在 X 点上的积分。若 X 为一列向量，Y 为矩阵，且 size(Y,1) = length(X)，则 trapz(X,Y)通过 Y 的第一个非单元集方向进行计算。
- T = trapz(…,dim)：沿着 dim 指定的方向对 Y 进行积分。若参量中包含 X，则应有 length(X)=size(Y,dim)。

【功能介绍】 用梯形法进行数值积分。

【实例 3.60】 用梯形法对函数 $y=\dfrac{5}{9+16x^2}$ 在区间 [−1 1] 上进行积分。

```
>>X = -1:0.1:1;
>>Y = 5/(9+16*X.^2);
>>T = trapz(X,Y)
```

计算结果为：

```
T =
0.5492
```

【实例讲解】 对上例中的算式用梯形法进行积分，并求出积分值。a、b 分别表示积分的上下限，积分步长规定为 0.1。

【实例 3.61】 用梯形法对函数 $y=\sin x$ 在区间 $[0,\pi]$ 上进行积分。

```
>> x=0:pi/100:pi;
>> y=sin(x);
>> k=trapz(x,y)
```

计算后的结果为：

```
k =
1.9998
```

【实例讲解】 $y=\sin x$ 在$[0,\pi]$上积分的值为 2，1.9998 近似等于 2。

3.4.4 rat、rats 函数——有理数近似求取

【背景知识】 虽然所有的浮点数值都是有理数，有时用简单的有理数字（分子与分母都是较小的整数）近似地表示它们是有必要的。函数 rat 将试图做到这一点。对于有连续出现的小数的数值，将会用有理式近似表示它们。函数 rats 调用函数 rat，且返回字符串。

【语法说明】

- [N,D] = rat(X) ：对于默认的误差，返回阵列 N 与 D，使 N./D 近似为 X。
- [N,D] = rat(X,tol) ：在指定的误差 tol 范围内，返回阵列 N 与 D，使 N./D 近似为 X。
- rat(X)、rat(X…) ：在没有输出参量时，简单地显示 x 的连续分数。
- S = rats(X,strlen) ：返回一包含简单形式的、X 中每一元素的有理近似字符串 S，若对于分配的空间中不能显示某一元素，则用星号表示。该元素与 X 中其他元素进行比较而言较小，但并非是可以忽略。参量 strlen 为函数 rats 中返回的字符串元素的长度。默认值为 strlen=13，这允许在 78 个空格中有 6 个元素。
- S = rats(X) ：返回与用 MATLAB 函数 format rat 显示 X 相同的结果给 S。

【功能介绍】 有理分式近似。

【实例 3.62】 对分式 $s=1-1/2+1/3-1/4+1/5-1/6+1/7$ 进行近似。

```
>>s = 1-1/2+1/3-1/4+1/5-1/6+1/7
>>format rat
>>S1 = rats(s)
>>S2 = rat(s)
>>[n,d] = rat(s)
>>PI1 = rats(pi)
>>PI2 = rat(pi)
```

计算结果为：

```
s =
   0.7595
S1 =
   319/420
S2 =
   1 + 1/(-4 + 1/(-6 + 1/(-3 + 1/(-5))))
n =
   319
d =
   420
PI1 =
   355/113
PI2 =
3 + 1/(7 + 1/(16))
```

【实例讲解】 s 是小数形式，S1 为分数形式，n、d 分别为分子和分母。

3.4.5 dblquad 函数——矩形区域二元函数重积分的计算

【语法说明】

- q = dblquad(fun,xmin,xmax,ymin,ymax) ：调用函数 quad 在区间[xmin,xmax，ymin,ymax]上计算二元函数 z=f(x,y)的二重积分。输入向量 x，标量 y，则 f(x,y)必须返回一个用于积分的向量。
- q = dblquad(fun,xmin,xmax,ymin,ymax,tol) ：用 tol 指定精度。
- q = dblquad(fun,xmin,xmax,ymin,ymax,tol,method) ：用指定的算法 method 代替默认算法 quad。method 的取值有@quadl 或用户指定的、与函数 quad 与 quadl 有相同调用次序的函数句柄。
- q = dblquad(fun,xmin,xmax, ymin,ymax,tol, method,p1, p2,…) ：将可选参数 p1,p2,..等传递给函数 fun(x,y,p1,p2,…)。若 tol=[]，method=[]，则使用默认精度和算法 quad。

【功能介绍】 矩形区域上的二重积分的数值计算。

【实例 3.63】 求函数在固定区域内的积分值。

```
>>fun = inline('y./sin(x)+x.*exp(y)');
>>Q = dblquad(fun,1,3,5,7)
```

计算结果为：

```
Q =
3.8319e+003
```

【实例讲解】 在横坐标为[1，3]和纵坐标为[5，7]区域内的积分值为 3831.9。

3.4.6 quad2dggen 函数——任意区域上二元函数的数值积分

【语法说明】

- q = quad2dggen(fun,xlower,xupper,ymin,ymax)：在由[xlower,xupper, ymin,ymax]指定的区域上计算二元函数 z=f(x,y)的二重积分。
- q = dblquad(fun,xlower,xupper,ymin,ymax,tol)：用 tol 指定精度，默认情况下精度为 10^{-6}，再进行计算。
- q = dblquad(fun,xmin,xmax,ymin,ymax,tol,method)：用指定的算法 method 代替默认算法。method 的取值有默认算法或用户指定的、与默认函数有相同调用次序的函数句柄。
- q=dblquad(fun,xlower,xupper,ymin,ymax, tol,method, p1,p2,…)：将可选参数 p1,p2,..等传递给函数 fun(x,y,p1,p2,…)。若 tol=[]，method=[]，则使用默认精度和算法。

【功能介绍】 任意区域上的二重积分的数值计算。

【实例 3.64】 计算单位圆域上的积分：$I=\iint\limits_{x^2+y^2\le 1} e^{-\frac{x^2}{2}}\sin(x^2+y)\mathrm{d}x\mathrm{d}y$

先把二重积分转化为二次积分的形式：$I=\int_{-1}^{1}\mathrm{d}y\int_{-\sqrt{1-y^2}}^{\sqrt{1-y^2}} e^{-\frac{x^2}{2}}\sin$

$(x^2+y)\mathrm{d}x$。

```
>>f = inline('exp(-x.^2/2).*sin(x.^2+y)','x','y');
>>xlower = inline('-sqrt(1-y.^2)','y'); xupper = inline('sqrt(1-y.^2)','y');
>>Q = quad2dggen(fun,xlower,xupper, -1,1,1e-4)
```

计算结果为：

```
Q =
    0.5368603818
```

【实例讲解】 读者可以利用微积分的知识进行验证。

3.4.7　diff 函数——微分函数

【语法说明】

- diff(f) ：返回 f 对预设独立变数的一次微分值。
- diff(f,'t')：返回 f 对独立变数 t 的一次微分值。
- diff(f,n) ：返回 f 对预设独立变数的 n 次微分值。
- diff(f, 't',n) ：返回 f 对独立变数 t 的 n 次微分值。

【功能介绍】 数值微分函数也是用 diff，因此这个函数是靠输入的引数决定是以数值或是符号微分，如果引数为向量则执行数值微分，如果引数为符号表示式则执行符号微分。

【实例 3.65】 先定义下列 3 个方程式，然后演算其微分项：

```
>>S1 = '6*x^3-4*x^2+b*x-5';
>>S2 = 'sin(a)';
>>S3 = '(1 - t^3)/(1 + t^4)';
>>diff(S1)
ans=18*x^2-8*x+b
>>diff(S1,2)
ans= 36*x-8
>>diff(S1,'b')
ans= x
>>diff(S2)
ans=
cos(a)
>>diff(S3)
```

```
ans=-3*t^2/(1+t^4) -4*(1-t^3)/(1+t^4)^2*t^3
>>simplify(diff(S3))
ans= t^2*(-3+t^4-4*t)/(1+t^4)^2
```

【实例讲解】 本例首先对第一个多项式 S1 求了一次导数和两次导数，然后分别对 S2 和 S3 求微分。

【实例3.66】 在区间[-3 6]上求函数 $y=1-9x+11x^2+23x^3-10x^4+35x^5$ 的微分。

```
>> x=linspace(-3,6);        % 产生100个x的离散点
>> p=[1 -9 11 23 -10 35];
>> f=polyval(p,x);
>> plot(x,f)                % 将多项式函数绘图
>> title('Fifth-deg. equation')
>> dfb=diff(f)./diff(x);       % 注意要分别计算 diff(f)和
diff(x)
>> xd=x(2:length(x));      % 注意只有 99 个 df 值,而且是对应
x2,x3,...,x100 的点
>> plot(xd,dfb )            % 将多项式的微分项绘图
>> title('Derivative of fifth-deg. equation')
```

得到的结果如图 3.36 和图 3.37 所示。

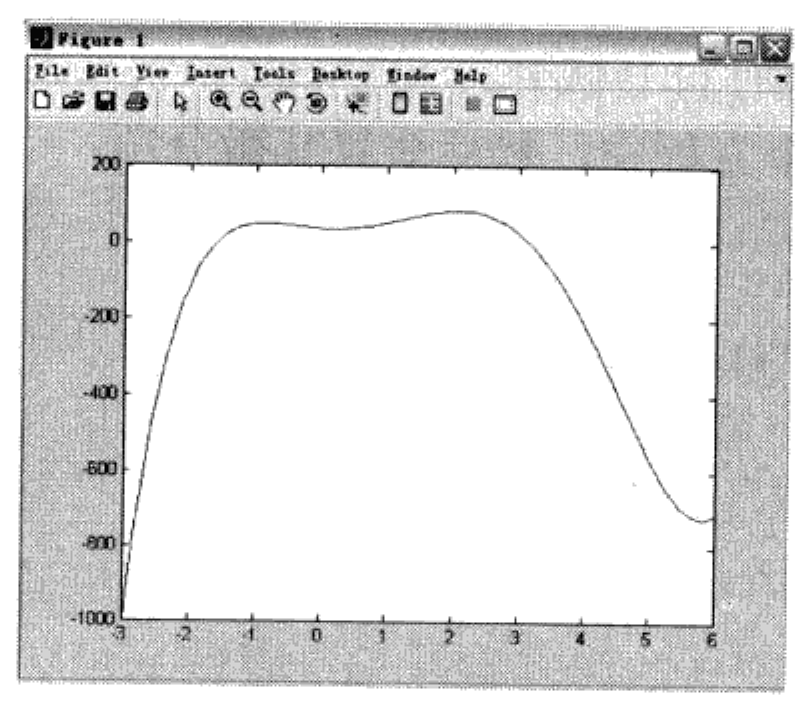


图 3.36 多项式图形

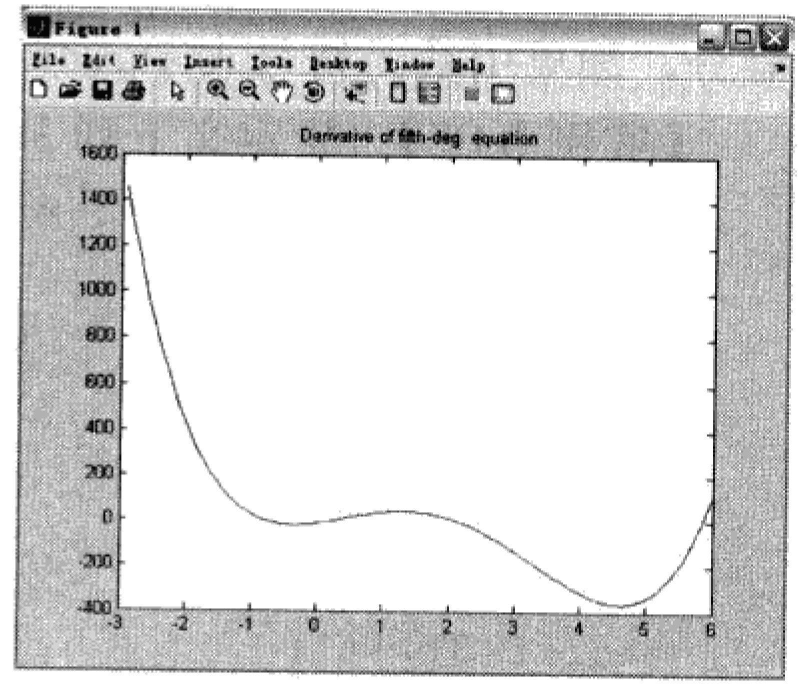


图 3.37 多项式的微分图形

【实例讲解】 读者可以对多项式进行微分运算，然后绘制对应的图形，进行对比。

【实例 3.67】 利用中央差分格式对上例中的函数进行微分计算。

```
>> x=linspace(-3,6); % 产生 100 个 x 的离散点
>> p=[1 -9 11 23 -10 35];
>> f=polyval(p,x);
>> plot(x,f) % 将多项式函数绘图
>> num=f(3:length(f)) -f(1:length(f) -2); % 中央差分是
f(k+1) -f(k-1)
>> deno=x(3:length(f)) -x(1:length(f) -2); % 中央差分是
x(k+1) -x(k-1)
>> df_c=num./deno;
>> xd=x(2:length(x) -1); % xd 的点数只有 98 个
>> plot(xd,df_c)
```

得到的结果如图 3.38 所示。

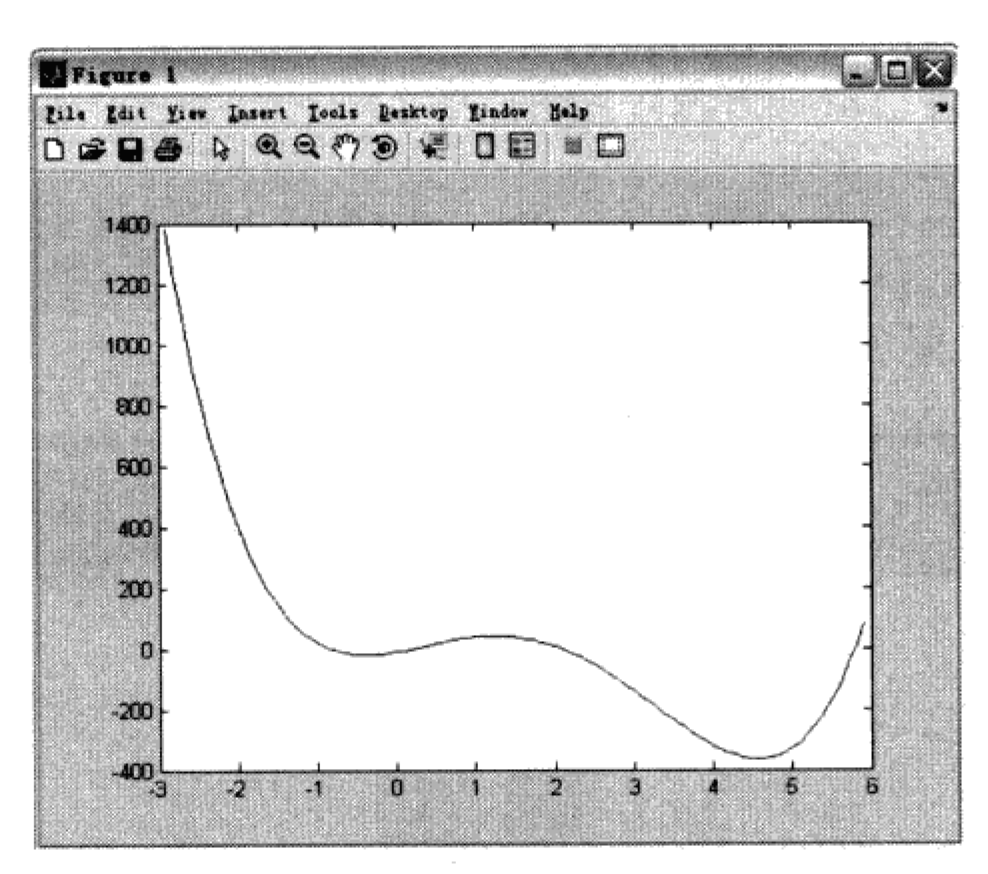


图 3.38　中央差分微分图

【实例讲解】 利用中央差分格式计算的话就不能直接用 diff 函数求取，而是要自行计算自变量和函数的两点之差。

【实例 3.68】 求离散点 y=[−.447 1.978 3.28 6.16 7.08 7.34 7.66 9.56 9.48 9.30 11.2]的微分。

```
>> x=0:0.1:1;
>> y=[-.447 1.978 3.28 6.16 7.08 7.34 7.66 9.56 9.48 9.30
11.2]
```

```
y =

  Columns 1 through 6

   -0.4470    1.9780    3.2800    6.1600    7.0800    7.3400

  Columns 7 through 11

    7.6600    9.5600    9.4800    9.3000   11.2000

>> plot(x,y,'o',x,y)
>> title('y(x) data plot')
>> ylabel('y(x)'), xlabel('x')
>> dy=diff(y)./diff(x);
>> xd=x(1:length(x) -1);
>> plot(xd,dy)
>> title('Approximate derivative using diff')
>> ylabel('dy/dx'), xlabel('x')
```

得到的结果如图 3.39 和图 3.40 所示。

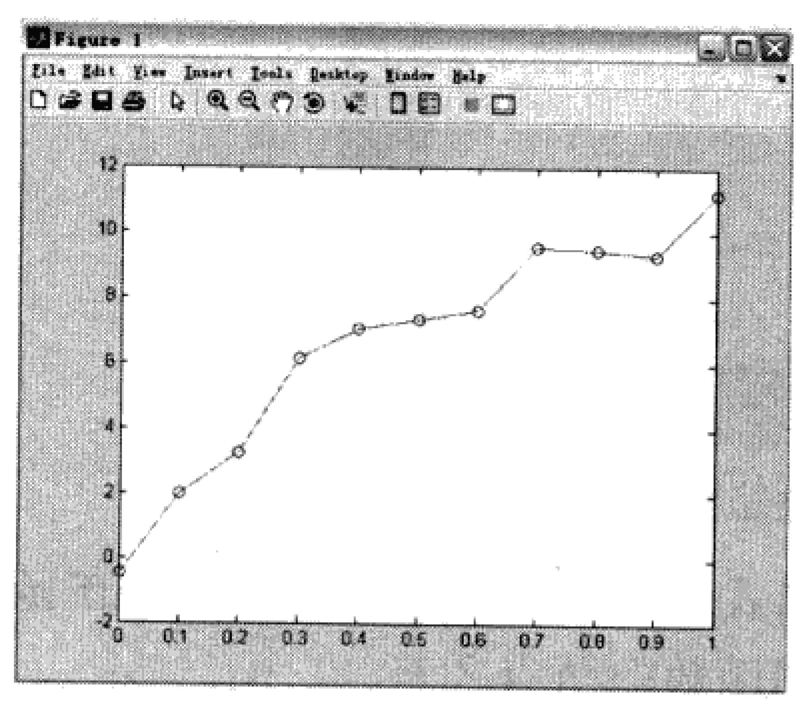



图 3.39　离散数据显示

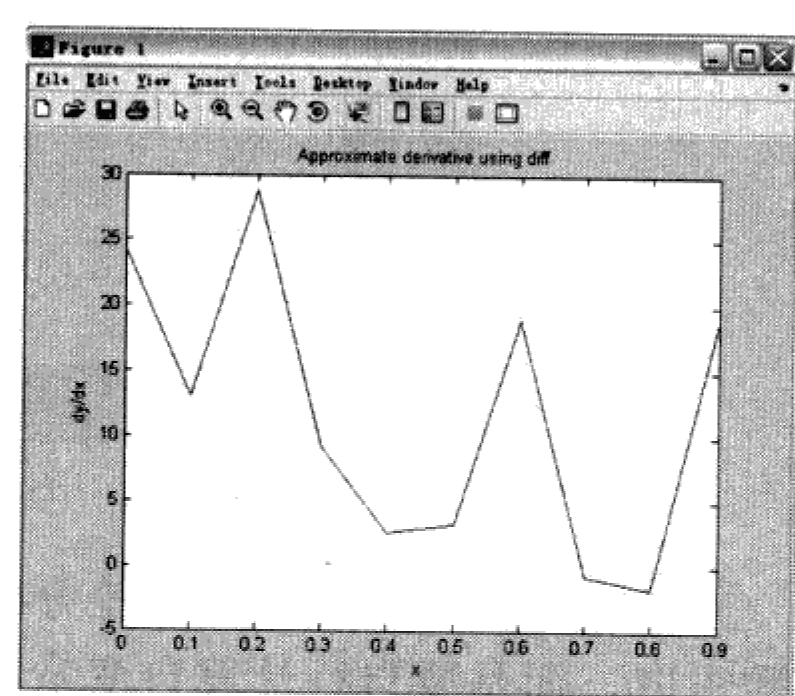


图 3.40　离散数据微分

【实例讲解】 以上的例子是针对数据组为离散形态，要注意的是原数据所代表的函数分布并无明显的折角，但是它的一阶微分经数值微分计算后的微分函数分布却有极大的曲折变化。

3.4.8 int 函数——积分函数

【语法说明】 相关的函数语法有下列 4 个。

- int(f) ：返回 f 对预设独立变量的积分值。
- int(f,'t') ：返回 f 对独立变量 t 的积分值。
- int(f,a,b) ：返回 f 对预设独立变量的积分值，积分区间为 [a,b]，a 和 b 为数值式。
- int(f,'t',a,b) ：返回 f 对独立变量 t 的积分值，积分区间为 [a,b]，a 和 b 为数值式。
- int(f,'m','n') ：返回 f 对预设变量的积分值，积分区间为 [m,n]，m 和 n 为符号式。

【功能介绍】 int 函数用以演算一函数的积分项，这个函数要找出一符号式 F 使得 diff(F)=f。如果积分式的解析式 (analytical form, closed form) 不存在的话或是 MATLAB 无法找到，则 int 传回原输入的符号式。

【实例 3.69】 先定义 3 个多项式，然后求其积分项。

```
>>S1 = '6*x^3-4*x^2+b*x-5';
>>S2 = 'sin(a)';
>>S3 = 'sqrt(x)';
>>int(S1)
ans= 3/2*x^4-4/3*x^3+1/2*b*x^2-5*x
>>int(S2)
ans= -cos(a)
>>int(S3)
ans= 2/3*x^(3/2)
>>int(S3,'a','b')
ans= 2/3*b^(3/2) - 2/3*a^(3/2)
>>int(S3,0.5,0.6)
ans= 2/25*15^(1/2) -1/6*2^(1/2)
```

【实例讲解】 这个例子分别对多项式和数学函数式进行积分运算，与 int(S1)形式相同的是不定积分，而 int(S3,0.5,0.6)是定积分，所以能够直接求出积分值。

3.4.9 roots 函数——求多项式的根

【语法说明】 r=roots(p)：p 为多项式的各阶的系数。

【功能介绍】 一个多项式视其阶数而定，它的根可以有一个或者几个，可能为实数也可能为复数。要求一高阶多项式的根往往需要借助数值方法，MATLAB 已将这些数值方法写成一函数 roots(p)，我们只要输入多项式的各阶系数（p）即可求解到对应的根。

【实例 3.70】 求多项式 $2x^2+3x+1$ 和多项式 $1-12x+25x^3+116x^4$ 的根。

```
>> p1=[1 3 2];
>> r1=roots(p)
r =
-2
-1
>> p2=[1 -12 0 25 116]; % 注意二阶项系数为零需要输入,否则多项式的阶数就不对
>> r=roots(p)                    % 有实数根及复数根
r 2=
11.7473
2.7028
-1.2251 + 1.4672i
-1.2251 - 1.4672i
```

【实例讲解】 第一个多项式只含有实数根，而第二个多项式即有实数根又有复数根。

3.4.10 poly 函数——通过根求原多项式

【语法说明】 poly(r)：r 代表根的阵列。

【功能介绍】 这个函数的用途是验算求解的根展开能求得原多项式。得到的数值为原多项式的系数向量。

【实例 3.71】 通过多项式的根来求解多项式。

```
>> r=[-2 1];
>> pp=poly(r)          % pp=(x+2)(x-1)=x^2+3x+2
```

```
pp =
1 3 2
>> p=[1 -4 6 -4];
>> r=roots(p)
r =
2.0000 1.0000 + 1.0000i 1.0000 - 1.0000i
>> pp=poly(r)        % 这个多项式的系数与原多项式 p 相同
pp =
1 -4 6 -4
>> pp=[1 7 12 9];    % 再看另一个多项式
>> r=roots(pp)
r =
-4.9395
-1.0303 + 0.8721i
-1.0303 - 0.8721i
>> pp=poly(r)      % 注意因计算的误差会有假虚部产生
pp =
1.0000 7.0000 12.0000 9.0000 + 0.0000i
```

【实例讲解】 通过这个例子可以看出，poly 与 roots 相当于互为逆运算。

3.4.11 real 函数——还原多项式

【语法说明】 real(r)：r 代表根的阵列。

【功能介绍】 这个函数的用途是验算求解的根展开能求得原多项式。得到的数值为原多项式的系数向量，但是 real 与 poly 的不同是前者能够将假虚部去掉，然后将多项式还原。

【实例 3.72】 还原多项式。

```
>> p=[1 -4 6 -4];
>> r=roots(p)
r =
2.0000 1.0000 + 1.0000i 1.0000 - 1.0000i
>> pp=poly(r) % 这个多项式的系数与原多项式 p 相同
pp =
1 -4 6 -4
>> pp=[1 7 12 9]; % 再看另一个多项式
```

```
>> r=roots(pp)
r =
-4.9395
-1.0303 + 0.8721i
-1.0303 - 0.8721i
>> pp=poly(r) % 注意因计算的误差会有假虚部产生
pp =
1.0000 7.0000 12.0000 9.0000 + 0.0000i
>> pp=real(pp) % 可以 real 将假虚部去除,将原多项式还原
pp =
1.0000 7.0000 12.0000 9.0000
```

【实例讲解】 通过最后的 real 函数可以看到，它将前面得到的结果的虚部已经清除掉，得到的数值全部为实数值。

3.4.12 dsolve 函数——求解常微分方程式

【语法说明】 MATLAB 解常微分方程式的语法是：

```
dsolve('equation','condition')
```

其中 equation 代表常微分方程式即 y'=g(x,y)，且须以 Dy 代表一阶微分项 y' D2y 代表二阶微分项 y"，condition 则为初始条件。

【功能介绍】 求解常微分方程。

【实例 3.73】 给定 3 个一阶常微分方程和初始条件，然后求出函数表达式。

```
y'=3x2, y(2)=0.5
y'=2.x.cos(y)2, y(0)=0.25
y'=3y+exp(2x), y(0)=3
对应上述常微分方程式的符号运算式为:
>>soln_1 = dsolve('Dy = 3*x^2','y(2)=0.5')
ans= x^3-7.500000000000000
>>ezplot(soln_1,[2,4])

>>soln_2 = dsolve('Dy = 2*x*cos(y)^2','y(0) = pi/4')
ans= atan(x^2+1)
>>soln_3 = dsolve('Dy = 3*y + exp(2*x)',' y(0) = 3')
ans= -exp(2*x)+4*exp(3*x)
```

【实例讲解】 按照上面的语法格式即可方便地求得函数表达式。

3.4.13　fzero 函数——求一元函数的零点

【语法说明】

- X=fzero(fun,x0)：参数 fun 表示的是一元函数，x0 表示求解的初始数值。
- [x,fval,exitflag,output]=fzero(fun,x0,options)：参数 options 表示优化迭代所采用的参数选项，在输出参数中，fval 表示对应的函数值，exitflag 表示程序退出的类型，output 则反映优化信息的变量。

【功能介绍】 求一元函数的零点。

要求任一方程式的根有 3 个步骤。

（1）先定义方程式。要注意必须将方程式安排成 $f(x)=0$ 的形态，例如一方程式为 sin(x)=3，则该方程式应表示为 $f(x)=\sin(x)-3$。

（2）代入适当范围的 $x, y(x)$ 值，将该函数的分布图画出。

（3）由图中决定 $y(x)$在何处附近($x0$)与 x 轴相交，如果从函数分布图看出根不只一个，则需再代入另一个在根附近的 $x0$，再求出下一个根。

【实例 3.74】 求方程 sinx=0 在 x=3、6、10 附近的根。

```
>> r=fzero('sin',3) % 因为 sin(x)是内建函数,其名称为 sin,因此无须定义它
r = % 选择 x=3 附近求根
3.1416
>> r=fzero('sin',6) % 选择 x=6 附近求根
r =
6.2832
>>  r=fzero('sin',10)


r =

   9.4248
```

【实例讲解】 通过 3 个结果知道，每一个都是π的倍数，而且

是最接近所给定值的根。

【实例 3.75】 不知道方程式的形态，不过可以将它划出来，再找出根的位置。

```
>> x=linspace(-3,5);
>> y=humps(x);
>> plot(x,y);
>>grid
>> r=fzero('humps',1.2)    %找出 1.2 附近的根

r =

1.2995
>>  r=fzero('humps',0.1)   %找出 0.1 附近的根

r =

  -0.1316
```

得到的结果如图 3.41 所示。

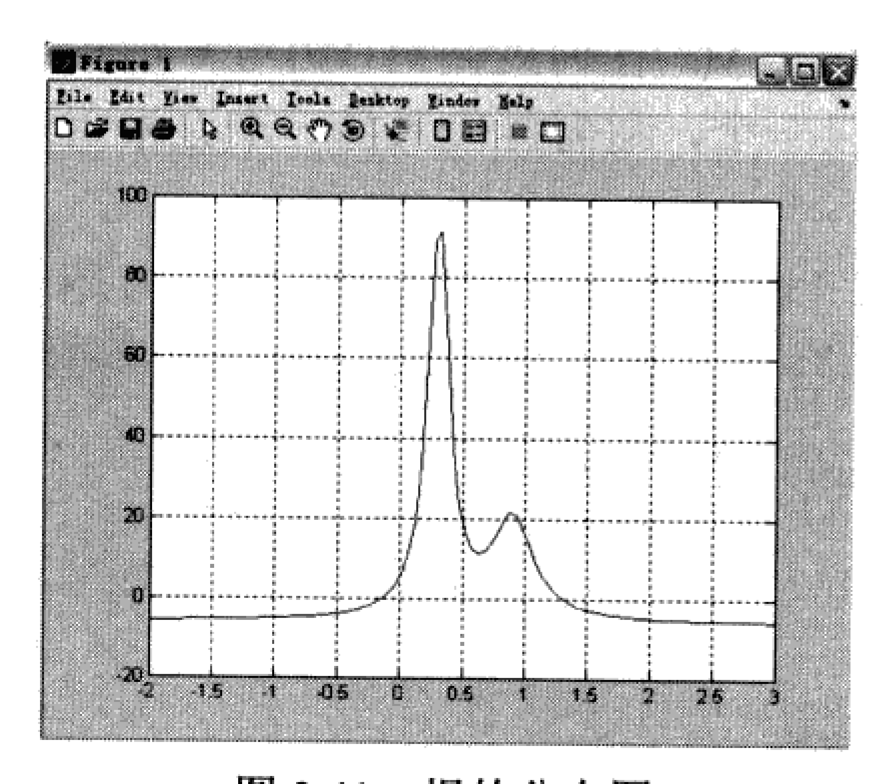


图 3.41 根的分布图

【实例讲解】 通过图形中根的分布很容易看出，在 1.2 附近和 0 附近的两个根的情况。

【实例 3.76】 先建立一个 M 文件，这个函数只用来求解一个函数值。

```
% m-function, f_1.m
function y=f_1(x) % 定义 f_1.m 函数
y=x.^3-2*x-5;
>> x=linspace(-2,3);
>> y=f_1(x);
>> plot(x,y), grid      % 由图中可看出在 2 附近有一个根
>> r=fzero('f_1',2);    % 决定在 2 附近的根
r =
2.0946
>> p=[1 0 -2 -5]
>> r=roots(p)           % 以求解多项式根方式验证
r =
2.0946
-1.0473 + 1.1359i
-1.0473 - 1.1359i
```

得到的结果如图 3.42 所示。

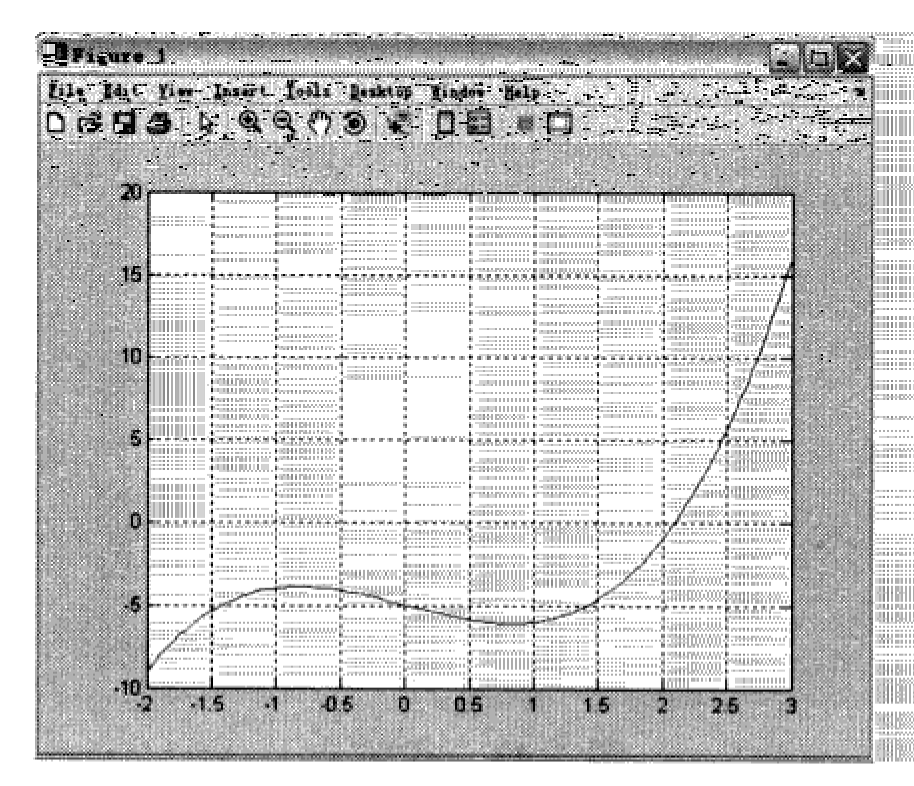


图 3.42　y=x.^3-2*x-5 的根的分布

【实例讲解】读者可以修改初始求解的数值，查看不同的结果。

【实例 3.77】 求函数 $y = x^2 \sin x - x + 1$ 在闭区间[-4，+4]上的零点。

```
>> %计算函数值
>> x=[-4:0.1:4];
>> y=sin(x).*x.^2-x+1;
>> plot(x,y,'r');
```

```
>> hold on
>> % 添加水平线
>> h=line([-4,4],[0,0]);
>> %设置直线的宽度和颜色
>> set(h,'LineWidth',2);
>> set(h,'Color','k');
>> %设置坐标轴刻度
>> set(gca,'Xtick',[ -4:0.5:4]);
>> %添加图形标题和坐标轴名称
>> title('求零点');
>> grid
>> xlabel('x')
>> ylabel('y')
```

得到的结果如图 3.43 所示。

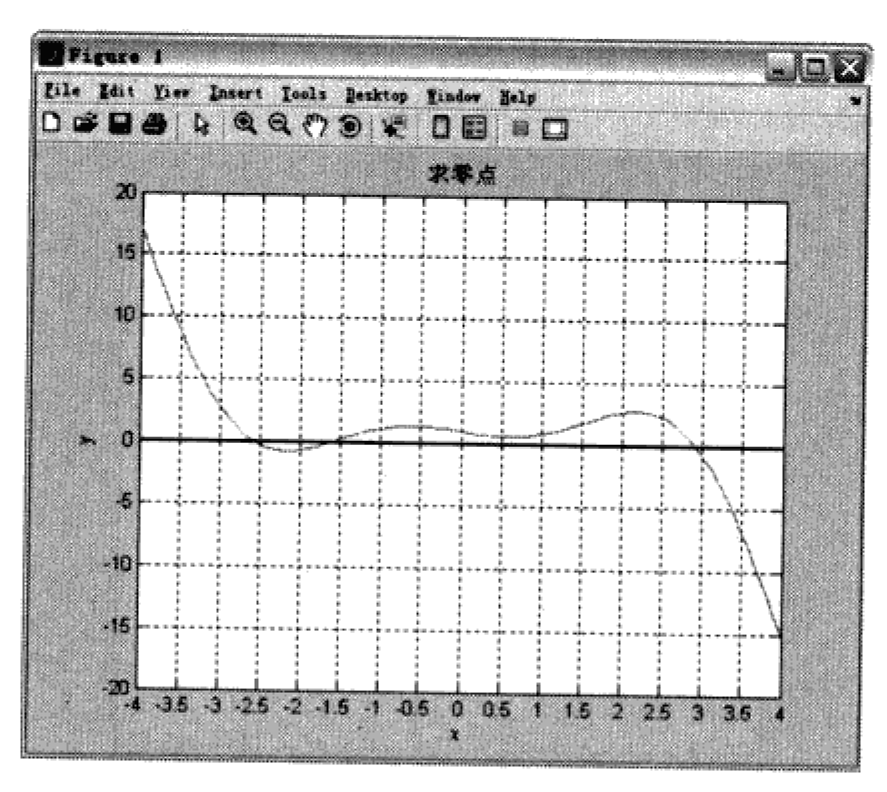


图 3.43 求零点

【实例讲解】 通过图形很容易求出这个函数的零点，这个函数在[−4，4]上共有 3 个零点。

3.4.14 龙格-库塔法解微分方程

【背景知识】 龙格-库塔（Runge-Kutta）方法是最通用的解 ODE 的方法，它可以依计算精确度的要求有低阶到高阶的各个计算式，因为一阶龙格-库塔法的精确度太低，所以在真正解 ODE 时，最少

需用二阶以上的方法。

【语法说明】

- ode23('dy',x0,xn,y0)：其中 dy 为 ODE 中的等式右边的函数，x0, xn 是要解 ODE 的区间[*x0, xn*]的二个端点，y0 是起始值(*y0*=*y*(*x0*))，同时以二阶及三阶阮奇-库达法求解。
- ode45('dy',x0,xn,y0)：各个参数与 ode23 的解释相同，它是以四阶及五阶阮奇-库达法求解。

【功能介绍】 解微分方程。

【实例 3.78】 求 $dy = 3x^2$ 的解。

```
% m-function, g1.m
function dy=g1(x,y)
dy=3*x.^2;
>> [x,num_y]=ode23('g1',2,4,0.5);
>> anl_y=x.^3-7.5;
>> plot(x,num_y,x,anl_y,'o')
>> title('Solution of g1')
>> xlabel('x'), ylabel('y=f(x)'), grid
```

得到的结果如图 3.44 所示。

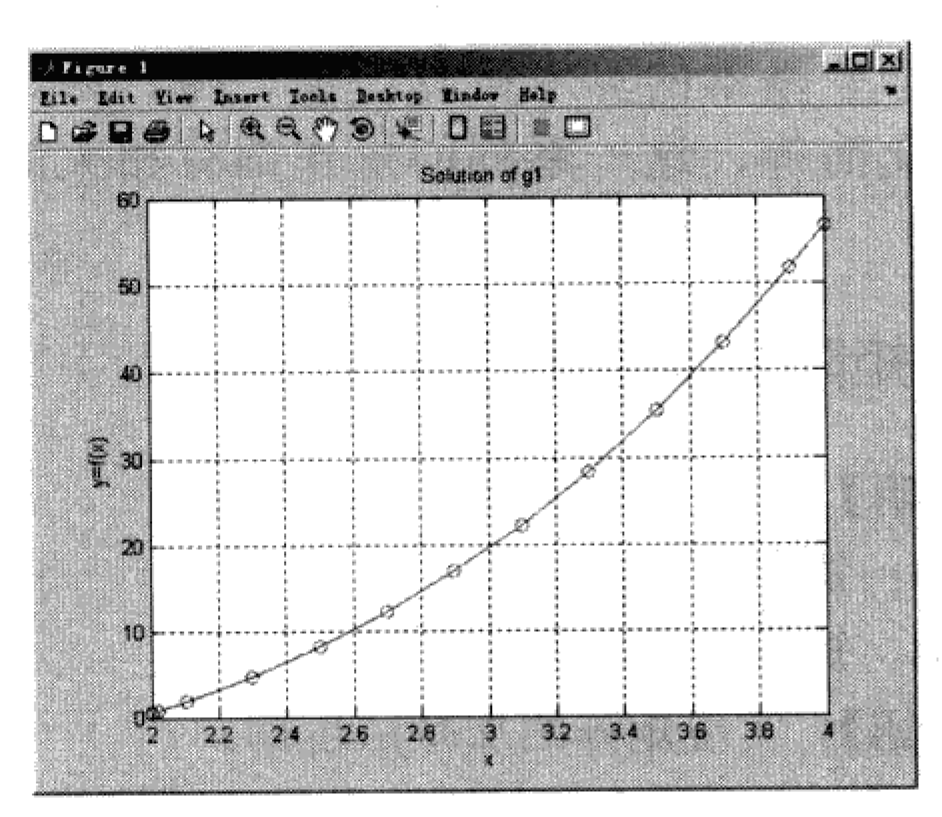
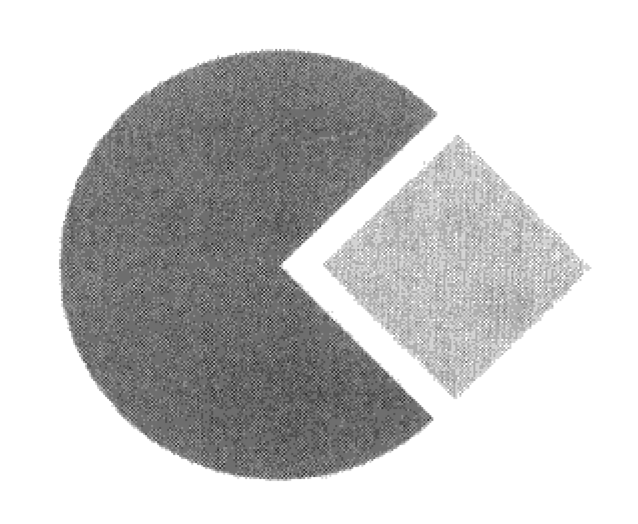


图 3.44　查看微分方程的解

【实例讲解】 读者可以利用微分方程的解析解来进行结果对比。



第 4 章　符号运算函数

符号运算是数值运算中较为常用的，MATLAB 中的符号数学工具箱适用于广泛的用途，它仅仅应用于一些特殊专业或专业分支。除此，MATLAB 符号数学工具箱与其他工具箱的不同之处还在于它使用字符串来进行符号分析，并非基于数组的数值分析。

4.1　算术符号运算

算术符号运算主要是围绕数值问题展开的，这一节包括了符号表达式的一些基本的算术运算，加、减、乘、除及合并、分解、展开、化简等，这一系列的操作都有相关的函数，下面一一介绍。

4.1.1　矩阵加减运算

【语法说明】

- A+B：矩阵的加法。
- A−B：矩阵的减法。

【功能介绍】 实现矩阵的加减法运算。

【实例 4.1】 定义两个符号矩阵并求加减运算。

```
>> syms a
>> syms b
>> a+b
ans =
a+b
>> a-b
ans =
a-b
```

【实例讲解】 从上面的结果可以看出，符号矩阵的运算和数值运算很类似。

4.1.2　符号矩阵乘法

【语法说明】

- A*B：符号矩阵乘法。
- A.*B：符号数组的乘法。

【功能介绍】 实现符号矩阵或数组相乘。

【实例 4.2】 实现两个符号矩阵的相乘。

```
>> syms a b c d
>> A=[a b;c d]
A =
[ a, b]
[ c, d]
>> syms a e f g
>> B=[a e;f g]
B =
[ a, e]
[ f, g]
>> A*B
ans =
[ a^2+b*f, a*e+b*g]
[ c*a+d*f, c*e+d*g]
```

【实例讲解】 数组的乘法与此相同，不做介绍。

4.1.3　符号除法运算

【语法说明】

- A\B：矩阵的左除法。

- A.\B：数组的左除法。

【功能介绍】 实现矩阵的乘除法运算。

【实例 4.3】 演示矩阵的乘除法计算。

```
>> syms a b c d
>> A=[a b;c d]
A =
[ a, b]
[ c, d]
>> syms a e f g
>> B=[a e;f g]
B =
[ a, e]
[ f, g]
 [ c*a+d*f, c*e+d*g]
>> A.\B
ans =
[   1, e/b]
[ f/c, g/d]
>> A\B
ans =
[    (-b*f+a*d)/(a*d-c*b),    (-b*g+e*d)/(a*d-c*b)]
[      a*(-c+f)/(a*d-c*b),  -1/(a*d-c*b)*(-a*g+c*e)]
```

【实例讲解】 第一步是将 A、B 作为数组相除得到的结果，而第二步 A、B 是作为矩阵相除的结果。

4.1.4 符号的转置运算

【语法说明】

- A'：矩阵的 Hermition 转置。
- A.'：数组转置。

【功能介绍】 实现符号矩阵或数组的转置运算。

【实例 4.4】 对上例中 A 进行转置操作。

```
>> A'
ans =
[ conj(a), conj(c)]
[ conj(b), conj(d)]
```

```
>> A.'
ans =
[ a, c]
[ b, d]
```

【实例讲解】 读者可以结合上面的例子，了解不同转置的差别。

4.1.5 符号的乘方运算

【语法说明】

- A^B：矩阵的方幂。
- A.^B：数组的方幂。

【功能介绍】 实现矩阵或数组的乘方运算。

【实例 4.5】 对上例中的矩阵或数组 A 进行 3 次方运算。

```
>> A^3
ans =
[ a*(a^2+c*b)+b*(c*a+d*c), a*(a*b+b*d)+b*(c*b+d^2)]
[ c*(a^2+c*b)+d*(c*a+d*c), c*(a*b+b*d)+d*(c*b+d^2)]
 >> A.^3
ans =
[ a^3, b^3]
[ c^3, d^3]
```

【实例讲解】 矩阵的方幂和数组的方幂得到的结果是截然不同的，数组的方幂就是原数组中的每个元素分别求方幂运算，而矩阵的运算则要符合矩阵运算的特点。

【实例 4.6】 综合应用符号的运算函数进行演示。

```
>>p1 = '1/(y-3)';
>>p2 = '3*y/(y+2)';
>>p3 = '(y+4)*(y-3)*y';
>> symmul(p1,p3)
ans=
(y+4)*y
>> sympow(p2,3)
ans=
27*y^3/(y+2)^3
>> symadd(p1,p2)
ans=
1/(y-3)+3*y/(y+2)
```

```
>>[num,den] = numden(symadd(p1,p2))
ans=
[-8*y+2+3*y, (y-3)*(y+2)]
>> horner(symadd(p3,'1')
ans=
1+(-12+(1+y)*y)*y
```

【实例讲解】 读者可以结合前面对各种函数的求解，来分析上面的例子。

4.1.6　size 函数——符号矩阵的维数

【语法说明】

- d=size(A)：若 A 为 m×n 阶的符号矩阵，则输出结果 d=[m，n]。
- [m,n] = size(A)：分别返回矩阵 A 的行数 m，列数 n。
- d= size(A, n)：返回由标量 n 指定的 A 的方向的维数：n=1 为行方向，n=2 为列方向。

【功能介绍】 返回符号矩阵的维数。

【实例 4.7】 创建符号矩阵，并求解矩阵的维数。

```
>>syms a b c d
>>A = [a b c ; a b d; d c b; c b a];
>>d = size(A)
>>r = size(A, 2)   %n=2 时表示返回列方向的维数
```

计算结果为：

```
d =
     4     3
r =
     3
```

【实例讲解】 使用 size 函数，用户可以求解各种层面的维数。

4.1.7　compose 函数——复合函数运算

【语法说明】

- compose(f,g)：返回复合函数 f[g(y)]，其中 f=f(x)，g=g(y)。其中符号 x 为函数 f 中由函数 findsym(f) 确定的符号变量，符号 y

为函数 g 中由函数 findsym(g) 确定的符号变量。

- compose(f,g,z)：返回复合函数 f[g(z)]，其中 f=f(x)，g=g(y)，符号 x、y 为函数 f、g 中由函数 findsym 确定的符号变量。
- compose(f,g,x,z)：返回复合函数 f[g(z)]，而令变量 x 为函数 f 中的自变量 f=f(x)。令 x=g(z)，再将 x=g(z)代入函数 f 中。
- compose(f,g,x,y,z)：返回复合函数 f[g(z)]。而令变量 x 为函数 f 中的自变量 f=f(x)，而令变量 y 为函数 g 中的自变量 g=g(y)。令 x=g(y)，再将 x=g(y)代入函数 f=f(x)中，得 f[g(y)]，最后用指定的变量 z 代替变量 y，得 f[g(z)]。

【功能介绍】 复合函数运算。

【实例 4.8】 对下面几个函数 f,g,h,p 进行函数替换操作。

```
>> syms x y z  u v w
>> f = 1/(2*z + x^2*y);
>>g = x^v;
>>h= sin(w);
>>p = sqrt(y/u);
>> X = compose(f,g)    % 令 x=g=x^v
X =
1/(2*z+(x^v)^2*y)
 >> Y = compose(f,g,v)    % 令 x=v
Y =
1/(2*z+(v^v)^2*y)
>> E = compose(h,g,x,z)
E =
sin(w)
>> F = compose(h,p,x,y,z)
F =
sin(w)
```

【实例讲解】 灵活使用 compose 函数，可以处理很多复杂的函数运算问题。

4.1.8 colspace 函数——返回列空间的基

【语法说明】 B=colspace(A)：返回矩阵 B，其列向量形成由矩阵 A 的列向量形成的空间的坐标基，其中 A 可以是符号或数值矩阵。

【功能介绍】 返回列空间的基。

【实例 4.9】 返回向量 $A=\begin{bmatrix}1 & a\\ 2 & b\\ 3 & c\end{bmatrix}$ 矩阵的列空间的基。

```
>>syms  a b c
>>A = sym([1,a;2,b;3,c])
>>B = colspace(A)
```

计算结果为：

```
A =
    [ 1, a]
    [ 2, b]
    [ 3, c]
B =
    [                1,              0]
    [                0,              1]
    [ -(3*b-2*c)/(-b+2*a),   (-c+3*a)/(-b+2*a)]
```

【实例讲解】 关于矩阵列空间的基，读者可以查看相应的图书。

4.1.9 real 函数——求符号复数的实数部分

【语法说明】 real(Z)：返回符号复数 z 的实数部分。

【功能介绍】 返回符号复数的实数部分。

【实例 4.10】 求 $Y=4+3i$ 的实数部分。

```
>> Y=4+3i
Y =
   4.0000 + 3.0000i
>> real(Y)
ans =
     4
```

【实例讲解】 $4+3i$ 的实数部分为 4，利用 MATLAB 函数得到的结果与其相同。

4.1.10 image 函数——求符号复数的虚数部分

【语法说明】 imag(Z)：返回符号复数 z 的虚数部分。

【功能介绍】 求符号复数的虚数部分。

4.1.11　symsum 函数——符号表达式求和

【语法说明】

- r = symsum(s)：对符号表达式 s 中的符号变量 k 从 0～k-1 求和。
- r = symsum(s,v)：对符号表达式 s 中指定的符号变量 v 从 0～v-1 求和。
- r = symsum(s,a,b)：对符号表达式 s 中的符号变量 k（由函数 findsym(s)确定的）从 a～b 求和。
- r = symsum(s,v,a,b)：对符号表达式 s 中指定的符号变量 v 从 a～b 求和。

【功能介绍】 符号表达式求和。

【实例 4.11】 创建符号对象，然后进行符号表达式求和。

```
>>syms k n x
>>Y1 = symsum(k^3)
>>Y2= symsum(k^2-k)
>>Y3 = symsum(sin(k*pi)/k,0,n)
>>Y4 = symsum(k^2,0,10)
>>Y5 = symsum(x^k/sym('k!'), k, 0,inf) %为使 k!通过 MATLAB
表达式的检验,必须把它作为一个符号表达式。
```

计算结果为：

```
Y1 =
    1/4*k^4-1/2*k^3+1/4*k^2
Y2 =
    1/3*k^3-k^2+2/3*k
Y3 =
     -1/2*sin(k*(n+1))/k+1/2*sin(k)/k/(cos(k)-1)*cos
(k*(n+1))-1/2*sin(k)/k/(cos(k)-1)
Y4 =
     385
Y5 =
     exp(x)
```

【实例讲解】 符号表达式的求和运算与数值中的运算相同，只不过以符号的形式表达。

4.1.12 collect 函数——合并同类项

【语法说明】

- R = collect(S)：对于多项式 S 中的每一个函数，collect(S)按默认变量 x 的次数合并系数。
- R = collect(S,v)：对于多项式 S 中的每一个函数，collect(S)按指定的变量 v 合并系数。

【功能介绍】 合并同类项。

【实例 4.12】 实现简单的合并同类项操作。

```
>> syms a b c
>> 3*a+4*b-8*a+6*c-3*b
ans =
-5*a+b+6*c
>> X = collect((exp(a)+3*a)*(b+2))
X =
(exp(a)+3*a)*b+2*exp(a)+6*a
```

【实例讲解】 合并后的同类项仍以符号函数的形式表示。

4.1.13 expand 函数——符号表达式展开

【语法说明】 R = expand(S)：对符号表达式 S 中每个因式的乘积进行展开计算。

【功能介绍】 用于符号表达式的展开。

【实例 4.13】 创建符号表达式，然后进行展开。

```
>>syms x y a b c t
>>X1 = expand((x-2)*(x-4)*(y-t))
>>X2 = expand(cos(x+y))
>>X3 = expand(exp((a+b)^3))
>>X4 = expand(log(a*b/sqrt(c)))
>>X5 = expand([sin(2*t), cos(2*t)])
```

计算结果为：

```
X1 =
    x^2*y-x^2*t-6*x*y+6*x*t+8*y-8*t
```

```
X2 =
    cos(x)*cos(y)-sin(x)*sin(y)
X3 =
    exp(a^3)*exp(a^2*b)^3*exp(a*b^2)^3*exp(b^3)
X4 =
    log(a*b/c^(1/2))
X5 =
    [ 2*sin(t)*cos(t),    2*cos(t)^2-1]
```

【实例讲解】 这个例子中对几个函数按数学运算的方式进行了展开。

4.1.14　factor 函数——符号因式分解

【语法说明】 factor(X)：参量 X 可以是正整数、符号表达式阵列或符号整数阵列。若 X 为一正整数，则 factor(X)返回 X 的质数分解式。若 X 为多项式或整数矩阵，则 factor(X)分解矩阵的每一元素。

【功能介绍】 对符号表达式进行因式分解。

【实例 4.14】 创建符号对象，并进行因式分解。

```
>>syms a b x y
>>F1 = factor(x^4-y^4)
>>F2 = factor([a^2-b^2, x^3+y^3])
>>F3 = factor(sym('12345678901234567890'))
```

计算结果为：

```
F1 =
     (x-y)*(x+y)*(x^2+y^2)
F2 =
    [(a-b)*(a+b),   (x+y)*(x^2-x*y+y^2)]
F3 =
    (2)*(3)^2*(5)*(101)*(3803)*(3607)*(27961)*(3541)
```

【实例讲解】 在上例中，符号表达式分解成几个因式之积的形式，对于一个较大的数值，则分解为几个简单数值的乘积形式。

4.1.15　simplify 函数——符号表达式的化简

【语法说明】 R = simplify(S)：S 是符号表达式。

【功能介绍】 化简符号表达式。

【实例 4.15】 将符号表达式 $\sin^4 x+\cos^4 x$ 、 $c\log\sqrt{a+b}$ 及符号向量 $[\frac{x^2+5x+6}{x+2},\sqrt{16}]$ 化简。

```
>>syms x a b c
>>R1 = simplify(sin(x)^4 + cos(x)^4)
>>R2 = simplify(exp(c*log(sqrt(a+b))))
>>S = [(x^2+5*x+6)/(x+2),sqrt(16)];
>>R3 = simplify(S)
```

计算结果为：

```
R1 =
     2*cos(x)^4+1-2*cos(x)^2
R2 =
     (a+b)^(1/2*c)
R3 =
     [ x+3,   4]
```

【实例讲解】 通过 simplify 函数得到的结果与数值运算中的相同。

4.1.16 numden 函数——符号表达式的分子与分母

【语法说明】 [N,D] = numden(A)：将符号或数值矩阵 A 中的每一元素转换成整系数多项式的有理式形式，其中分子与分母是相对互素的。输出的参量 N 为分子的符号矩阵，输出的参量 D 为分母的符号矩阵。

【功能介绍】 求符号表达式的分子与分母。

【实例 4.16】 创建符号对象，并求解符号表达式的分子和分母。

```
>>syms x y a b c d;
>>[n1,d1] = numden(sym(sin(4/5)))
>>[n2,d2] = numden(x/y + y/x)
>>A = [a, 1/b;1/c d];
>>[n3,d3] = numden(A)
```

计算结果为：

```
n1 =
     6461369247334093
```

```
d1 =
    9007199254740992
n2 =
    x^2+y^2
d2 =
   y*x
n3 =
   [ a, 1]
   [ 1, d]
d3 =
   [ 1, b]
   [ c, 1]
```

【实例讲解】 这个例子中，n 对应的是分数的分子部分，d 对应分数的分母部分，只有两个同时才有意义。

4.1.17 double 函数——将符号矩阵转化为浮点型数值

【语法说明】 R = double(S)：将符号对象 S 转换为数值对象 R。若 S 为符号常数或表达式常数，double 返回 S 的双精度浮点数值表示形式；若 S 为每一元素是符号常数或表达式常数的符号矩阵，double 返回 S 每一元素的双精度浮点数值表示的数值矩阵 R。

【功能介绍】 将符号对象转化为浮点型数值对象。

【实例 4.17】 符号矩阵转化为数值矩阵及数值运算。

```
>>gold_ratio = double(sym('(sqrt(5)-1)/2'))     % 计算黄金分割率
>>T = sym(hilb(4))
>>R = double(T)
```

计算结果为：

```
gold_ratio =
        0.6180
T =
   [  1, 1/2, 1/3, 1/4]
   [ 1/2, 1/3, 1/4, 1/5]
   [ 1/3, 1/4, 1/5, 1/6]
   [ 1/4, 1/5, 1/6, 1/7]
R =
   1.0000    0.5000    0.3333    0.2500
```

```
        0.5000    0.3333    0.2500    0.2000
        0.3333    0.2500    0.2000    0.1667
        0.2500    0.2000    0.1667    0.1429
```

【实例讲解】 通过这个例子可以看出，经过化简后的表示形式更加直观，都是以数值的形式表示。

4.1.18 solve 函数——代数方程的符号解析解

【语法说明】

- g = solve(eq)：输入参量 eq 可以是符号表达式或字符串。若 eq 是一符号表达式 x^2 –2×x-1 或一没有等号的字符串“x^2-2×x-1”，则solve(eq)对方程eq中的默认变量（由函数findsym(eq)确定的变量）求解方程 eq=0。若输出参量 g 为单一变量，则对于有多重解的非线性方程，g 为一行向量。
- g = solve(eq,var)：对符号表达式或没有等号的字符串 eq 中指定的变量 var 求解方程 eq(var)=0。
- g = solve(eq1,eq2,…,eqn)：输入参量 eq1,eq2,…,eqn 可以是符号表达式或字符串。该函数对方程组 eq1,eq2,…,eqn 中由函数 findsym 确定的 n 个变量如 x1,x2,…,xn 求解。若 g 为一单个变量，则 g 为一包含 n 个解的结构；若 g 为有 n 个变量的向量，则分别返回结果给相应的变量。
- g = solve(eq1,eq2,…,eqn,var1,var2,…,varn)：对方程组 eq1,eq2,…,eqn 中指定的 n 个变量如 var1,var2,…,varn 求解。

【功能介绍】 求代数方程的符号解析解。

【实例 4.18】 求解符号形式的方程组。

```
>solve('a*x^2 + b*x + c')
>>solve('a*x^2 + b*x + c','b')
>>solve('x + y = 1','x - 11*y = 5')
>>A = solve('a*u^2 + v^2', 'u - v = 1', 'a^2 - 5*a +6')
```

计算结果为：

```
ans =
[ 1/2/a*(-b+(b^2-4*a*c)^(1/2))]
```

```
[ 1/2/a*(-b-(b^2-4*a*c)^(1/2))]
ans =
-(a*x^2+c)/x
ans =
    x: [1x1 sym]
    y: [1x1 sym]
A =
    a: [4x1 sym]
    u: [4x1 sym]
    v: [4x1 sym]
```

【实例讲解】 从上面的结果中可以看出，二元一次方程组的通解符合结果。

【实例 4.19】 先定义几个符号方程，并对其进行求解。

```
>>eq1 = 'x-3=4';      %也可写成'eq1=x-7'
>>eq2 = 'x*2-x-6=0';  % 也可写成'eq2=x*2-x-6'
>>eq3 = 'x2+2*x+4=0';
>>eq4 = '3*x+2*y-z=10';
>>eq5 = '-x+3*y+2*z=5';
>>eq6 = 'x-y-z=-1';
>>solve(eq1)
ans=
7
>>solve(eq2)
ans=
[[3],[-2]]'                    % 原方程式有二个根 3, -2
>>solve(eq3)
ans=
[[-1+I*3^(1/2)],[-1-I*3^(1/2)]]'%注意实根和虚根的表示式
>>solve(eq4,eq5,eq6)    % 解三个联立方程式
ans=
x = -2, y = 5, z = -6
```

【实例讲解】 读者可以使用数值矩阵的方法来求解这个方程组，然后将结果和上面的结果进行对比。

4.1.19 simple 函数——求符号表达式的最简形式

【语法说明】

- r = simple(S)：该函数试图找出符号表达式 S 的代数上的简

单形式，显示任意的能使表达式 S 长度变短的表达式，且返回其中最短的一个。若 S 为一矩阵，则结果为整个矩阵的最短形式，而不是每一个元素的最简形式。若没有输出参量 r，则该函数将显示所有可能使用的算法与表达式，同时返回最短的一个。

- [r,how] = simple(S)：没有显示中间的化简结果，但返回能找到的最短的一个。输出参量 r 为一符号，how 为一字符串，用于表示算法。

【功能介绍】 返回符号表达式的最简形式。

【实例 4.20】 化简下面几个符号表达式。

```
>>syms x
>>R1 = simple(cos(x)^4+sin(x)^4)
>>R2 = simple(2*cos(x)^2-sin(x)^2)
>>R3 = simple(cos(x)^2-sin(x)^2)
>>R4 = simple(cos(x)+(-sin(x)^2)^(1/2))
>>R5 = simple(cos(x)+i*sin(x))
>>R6 = simple((x+1)*x*(x-1))
>>R7 = simple(x^3+3*x^2+3*x+1)
```

计算的结果为：

```
R1 =
    1/4*cos(4*x)+3/4
R2 =
    3*cos(x)^2-1
R3 =
    cos(2*x)
R4 =
    cos(x)+i*sin(x)
R5 =
    exp(i*x)
R6 =
    x ^3-x
R7 =
    (x+1)^3
R8 =
    4*x^3-3*x
```

【实例讲解】 化简后的结果与数学中相同。

4.1.20 finverse 函数——函数的反函数

【语法说明】

- g = finverse(f)：返回函数 f 的反函数。其中 f 为单值的一元数学函数，如 $f=f(x)$。若 f 的反函数存在，设为 g，则有 $g[f(x)] = x$。
- g = finverse(f,u)：若符号函数 f 中有几个符号变量时，对指定的符号自变量 v 计算其反函数。若其反函数存在，设为 g，则有 $g[f(v)] = v$。

【功能介绍】 求函数的反函数。

【实例 4.21】 求以下两个简单符号函数的反函数。

```
>>syms x p q u v;
>>V1 = finverse(1/((x^2+p)*(x^2+q)))
>>V2 = finverse(exp(u-2*v),u)
```

计算结果为：

```
Warning: finverse(1/(x^2+p)/(x^2+q)) is not unique.
> In D:\MATLABR12\toolbox\symbolic\@sym\finverse.m
at line 43
V1 =
      1/2/x*2^(1/2)*(x*(-x*q-x*p+(x^2*q^2-2*x^2*q*p
+x^2*p^2+4*x)^(1/2)))^(1/2)
V2 =
      2*v+log(u)
```

【实例讲解】 在这个例子中，由于 V1 的反函数不惟一，所以 MATLAB 会自动给出警告信息。

4.1.21 poly 函数——求特征多项式

【语法说明】 p = poly(A)或 p = poly(A, v)：若 A 为一个数值阵列，则返回矩阵 A 的特征多项式的系数。若 A 为一个符号矩阵，则返回矩阵 A 的变量为 x 的特征多项式。若带上参量 v，则返回变量为 v 的特征多项式。

【功能介绍】 求特征多项式。

【实例 4.22】 创建数值矩阵，然后求解特征多项式。

```
>>A = hilb(4);
>>p = poly(A)
>>q = poly(sym(A))
>>s = poly(sym(A),z)
```

计算结果为：

```
p =
    1.0000   -1.6762    0.2652   -0.0017    0.0000
q =
    x^4-176/105*x^3+3341/12600*x^2-41/23625*x+1/6048000
s =
   -176/105*z^3+3341/12600*z^2-41/23625*z+1/6048000+z^4
```

【实例讲解】 从上面的结果可以看出，使用 poly 函数求解的结果是类似的，只是表达形式不同。

4.1.22 poly2sym 函数——将多项式系数向量转化为带符号变量的多项式

【语法说明】

- r = poly2sym(c)：将系数在数值向量 c 中的多项式转化成相应的带符号变量的多项式（按次数的降幂排列）。默认的符号变量为 x。
- r = poly2sym(c, v)：符号变量用 v 显示。poly2sym 使用函数 sym 的默认转换模式（有理形式）将数值型系数转换为符号常数。

【功能介绍】 多项式系数向量转化为带符号变量的多项式。

【实例 4.23】 多项式系数向量转化为带符号变量的多项式。

```
>>r1 = poly2sym([1 2 3 4])
>>r2 = poly2sym([.694228, sqrt(2), sin(pi/3)])
>>r3 = poly2sym([1 0 1 -1 2], y)
```

计算结果为：

```
  r1 =
      x^3+2*x^2+3*x+4
  r2 =
```

```
        6253049924220329/9007199254740992*x^2+x*2^(1/2)
+1/2*3^(1/2)
    r3 =
        y^4+y^2-y+2
```

【实例讲解】 完成了多项式系数向符号多项式的转变。

4.1.23　findsym 函数——从一符号表达式中或矩阵中找出符号变量

【语法说明】

- r = findsym(S)：以字母表的顺序返回表达式 S 中的所有符号变量（注：符号变量为由字母（除了 i 与 j）与数字构成的、字母打头的字符串）。若 S 中没有任何的符号变量，则 findsym 返回一空字符串。
- r = findsym(S,n)：返回字母表中接近 x 的 n 个符号变量。

【功能介绍】 从一个符号矩阵中找出符号变量。

【实例 4.24】 从下面的符号矩阵中找出符号变量。

```
>>syms a x y z t alpha beta
>>1 = findsym(sin(pi*t*alpha+beta))
>>S2 = findsym(x+i*y-j*z+eps-nan)
>>S3 = findsym(a+y,pi)
```

计算结果为；

```
S1 =
    pi, alpha, beta, t
S2 =
    NaN, x, y, z
S3 =
    a, y
```

【实例讲解】 当符号函数的反函数不惟一时，系统会给出警告信息。

4.1.24　horner 函数——嵌套形式的多项式的表达式

【语法说明】 R = horner(P)：若 P 为一个符号多项式的矩阵，该函数将矩阵的每一元素转换成嵌套形式的表达式 R。

【功能介绍】 表达嵌套形式的多项式。

【实例 4.25】 求解嵌套形式的多项式。

```
>>syms x y
>>H1 = horner(2*x^4-6*x^3+9*x^2-6*x-4)
>>H2 = horner([x^2+x*y;y^3-2*y])
```

计算结果为：

```
H1 =
    -4+(-6+(9+(-6+2*x)*x)*x)*x
H2 =
    [  x^2+x*y]
    [ (-2+y^2)*y]
```

【实例讲解】 从上面的结果中可以看出，当表达式中有多个变量的时候，MATLAB 会自动按照不同的变量进行整理。

4.2 符号函数求微积分

符号函数也同普通的数值运算相同，存在常用的微积分运算函数，下面的小节中将简要介绍。

4.2.1 limit 函数——求极限

【语法说明】

- limit(F,x,a)：计算当 x→a 时符号表达式 F=F(x)的极限值。
- limit(F,a)：用函数 findsym(F)确定 F 中的自变量，设为变量 x，再计算当 x→a 时 F 的极限值。
- limit(F)：用函数 findsym(F)确定 F 中的自变量，设为变量 x，再计算当 x→0 时 F 的极限值。
- limit(F,x,a,'right')或 limit(F,x,a,'left')：计算符号函数 F 的单侧极限，左极限 $x \to a^-$ 或右极限 x→a+。

【功能介绍】 对符号函数求极限。

【实例 4.26】 对符号多项式求解极限。

```
>>syms x a t h n;
>>L1 = limit((cos(x)-1)/x)
```

```
>>L2 = limit(1/x^2,x,0,'right')
>>L3 = limit(1/x,x,0,'left')
>>L4 = limit((log(x+h)-log(x))/h,h,0)
>>v = [(1+a/x)^x, exp(-x)];
>>L5 = limit(v,x,inf,'left')
>>L6 = limit((1+2/n)^(3*n),n,inf)
```

计算结果为：

```
L1 =
     0
L2 =
     inf
L3 =
     -inf
L4 =
     1/x
L5 =
     [ exp(a),      0]
L6 =
     exp(6)
```

【实例讲解】 L1 计算的是自变量 x->0 时函数(cos(x)-1)/1 的极限，结果为 0；L2 是当自变量 x 从右侧->0 时 $\frac{1}{x^2}$ 的极限，为 $+\infty$；L3 与其相反；L6 为当 n->∞ 时表达式 $\left(1+\frac{2}{n}\right)^{3n}$ 的极限。

4.2.2 diff 函数——符号函数导数求解

【语法说明】

- diff(S,'v')、diff(S,sym('v'))：对表达式 S 中指定符号变量 v 计算 S 的 1 阶导数。
- diff(S)：对表达式 S 中的符号变量 v 计算 S 的 1 阶导数，其中 v=findsym(S)。
- diff(S,n)：对表达式 S 中的符号变量 v 计算 S 的 n 阶导数，其中 v=findsym(S)。
- diff(S,'v',n)：对表达式 S 中指定的符号变量 v 计算 S 的 n

阶导数。

【功能介绍】 对符号函数求导。

【实例 4.27】 分别求解符号函数 $\sin x^2 y^2$ 的一阶导数和二阶导数，并求导数值。

```
>>syms x y t
>>D1 = diff(sin(x^2)*y^2,2)    %计算 ∂²/∂x² y² sin x²
>>D2 = diff(D1,y)              %计算 ∂/∂y(∂²/∂x² y² sin x²)
>>D3 = diff(t^6,6)
```

计算结果为：

```
D1 =
     -4*sin(x^2)*x^2*y^2+2*cos(x^2)*y^2
D2 =
     -8*sin(x^2)*x^2*y+4*cos(x^2)*y
D3 =
     720
```

【实例讲解】 如果原始的符号表达式比较复杂，经过 diff 函数求解得到的结果会比较复杂。

【实例 4.28】 求下列符号函数 $y = 6x^3 - 4x^2 + bx - 5$，$y = \sin a$ 和 $y = \dfrac{1-t^3}{1+t^4}$ 的导数。

```
>>S1 = '6*x^3-4*x^2+b*x-5';
>>S2 = 'sin(a)';
>>S3 = '(1 - t^3)/(1 + t^4)';
>>diff(S1)

ans=
18*x^2-8*x+b
>>diff(S1,2)

ans=
36*x-8
>>diff(S1,'b')

ans=
```

```
x
>>diff(S2)

ans=
cos(a)
>>diff(S3)

ans=
-3*t^2/(1+t^4)-4*(1-t^3)/(1+t^4)^2*t^3
>>simplify(diff(S3))
ans=
t^2*(-3+t^4-4*t)/(1+t^4)^2
```

【实例讲解】 通过验证可以知道上面 3 个函数的导数计算是正确的。

4.2.3 int 函数——符号函数的积分

【语法说明】

- R = int(S,v)：对符号表达式 S 中指定的符号变量 v 计算不定积分。需要注意的是，表达式 R 只是函数 S 的一个原函数，后面没有带任意常数 C。
- R = int(S)：对符号表达式 S 中的符号变量 v 计算不定积分，其中 v=findsym(S)。
- R = int(S,v,a,b)：对表达式 S 中指定的符号变量 v 计算从 a 到 b 的定积分。
- R = int(S,a,b)：对符号表达式 S 中的符号变量 v 计算从 a 到 b 的定积分，其中 v=findsym(S)。

【功能介绍】 求符号函数的积分。

【实例 4.29】 求下列几个符号函数的积分。

```
>>syms x z t alpha
>>T1 = int(-2*x/(1+x^3)^2)
>>T2 = int(x/(1+z^2),z)
>>T3 = int(INT2,x)
>>T4 = int(x*log(1+x),0,1)
>>T5 = int(2*x, sin(t), 1)
>>T6 = int([exp(t),exp(alpha*t)])
```

计算结果为：

```
T1 =
        -2/9/(x+1)+2/9*log(x+1)-1/9*log(x^2-x+1)-2/9*3^
(1/2)*atan(1/3*(2*x-1)*… 3^(1/2))-2/9*(2*x-1)/(x^2-x+1)
T2 =
        x*atan(z)
T3 =
        1/2*x^2*atan(z)
T4 =
        1/4
T5 =
        1-sin(t)^2
T6 =
        [ exp(t), 1/alpha*exp(alpha*t)]
```

【实例讲解】 利用 int 函数在给定积分区间的情况下直接求出数值，没有设定区间时给出积分后的函数表达式。

【实例 4.30】 求 $y = 6x^3 - 4x^2 + bx - 5$、$y = \sin a$ 和函数 $y = \sqrt{x}$ 的积分函数，即其函数的广义积分。

```
>>S1 = '6*x^3-4*x^2+b*x-5';
>>S2 = 'sin(a)';
>>S3 = 'sqrt(x)';
>>int(S1)
ans=
3/2*x^4-4/3*x^3+1/2*b*x^2-5*x
>>int(S2)
ans=
-cos(a)
>>int(S3)
ans=
2/3*x^(3/2)
>>int(S3,'a','b')
ans=
2/3*b^(3/2)- 2/3*a^(3/2)
>>int(S3,0.5,0.6)
ans=
2/25*15^(1/2)-1/6*2^(1/2)
>>numeric(int(S3,0.5,0.6)) % 使用 numeric 函数可以计算积
分的数值
```

```
ans=
0.0741
```

【实例讲解】 从上面的例子可以看出，可以使用 numeric 函数来计算符号运算的结果。

4.2.4 dsolve 函数——常微分方程的符号解

【语法说明】 r = dsolve('eq1,eq2,⋯','cond1,cond2,⋯','v')

【功能介绍】 对给定的常微分方程（组）eq1,eq2,...中指定的符号自变量 v，与给定的边界条件和初始条件 cond1,cond2,⋯求符号解（即解析解）r；若没有指定变量 v，则缺省变量为 t。

【实例 4.31】 设置不同的条件，求解微分方程组。

```
>>D1 = dsolve('D2y - Dy =exp(x)')
>>D2 = dsolve('t*D2f = Df*log((Dy)/t)')
>>D3 = dsolve('(Dy)^2 + y^2 = 1','s')
>>D4 = dsolve('Dy = a*y', 'y(0) = b')  % 带一个定解条件
>>D5 = dsolve('D2y = -a^2*y', 'y(0) = 1', 'Dy(pi/a) =
0')   % 带两个定解条件
>>[x,y] = dsolve('Dx = y', 'Dy = -x')%求解线性微分方程组
>>[u,v] = dsolve('Du=u+v,Dv=u-v')
```

计算结果为：

```
D1 =
    -exp(x)*t+C1+C2*exp(t)
D2 =
    y(t)=Int(exp(t*diff(f(t), '$' (t,2))/diff(f(t),t))
*t,t)+C1
D3 =
    [        -1]
    [         1]
    [  sin(s-C1)]
    [ -sin(s-C1)]
D4 =
    b*exp(a*t)
D5 =
    cos(a*t)
x =
    cos(t)*C1+sin(t)*C2
y =
```

```
      -sin(t)*C1+cos(t)*C2
   u =
      1/2*C1*exp(2^(1/2)*t) - 1/4*C1*2^(1/2)*exp(-2^(1/2)
*t) + 1/4*C1*2^(1/2) *exp (2^(1/2)*t) + 1/2*C1*exp(-2^
(1/2)*t) - 1/4*C2*2^(1/2)*exp(-2^(1/2)*t) +  1/4*C2 *2^
(1/2)*exp(2^(1/2)*t)
   v =
     -1/4*C1*2^(1/2)*exp(-2^(1/2)*t)+1/4*C1*2^(1/2)*exp
(2^(1/2)*t)+1/2*C2*exp
          (2^(1/2)*t)+1/4*C2*2^(1/2)*exp(-2^(1/2)*t)-
1/4*C2*2^(1/2)*exp(2^(1/2)*t)+  1/2*C2*exp(-2^(1/2)*t)
```

【实例讲解】 利用这个函数，直接求出常微分方程的解，如果像D4、D5带定解条件的会给出带一个常数或者不带常数的表达式。

【实例4.32】 假如有下面3个微分方程。

$$y'=3x^2, y(2)=0.5$$

$$y'=2\cdot x\cdot \cos(y)^2, y(0)=0.25$$

$$y'=3y+\exp(2x), y(0)=3$$

求解其解析函数。

```
>>y1 = dsolve('Dy = 3*x^2','y(2)=0.5')
ans=
x^3-7.500000000000000
>>y2 = dsolve('Dy = 2*x*cos(y)^2','y(0) = pi/4')
ans=
atan(x^2+1)
>>y3 = dsolve('Dy = 3*y + exp(2*x)',' y(0) = 3')
ans=
-exp(2*x)+4*exp(3*x)
```

【实例讲解】 读者可以将求解出来的结果，代入方程组中进行检验。

4.3 符号函数的作图

通常工程中用到的符号函数都是以表达式的形式体现，这样有利

于计算。但是，如果有时需要直观地表示符号函数的物理意义，就需要用图形来实现，这一节将要讲解有关符号函数作图的相关函数。

4.3.1　ezplot 函数——画符号函数的图形

【语法说明】

- ezplot(f)：对于显式函数 f=f(x)，在默认的范围[$-\pi<x<\pi$]上画函数 f(x)；对于隐函数 f=f(x,y)，在默认的平面区域[$-2\pi<x<2\pi$，$-2\pi<y<2\pi$]上画函数 f(x,y)的图形。
- ezplot(f,[min,max])：在指定的范围[min<x<max]内画函数表达式 f=f(x)。若没有图形窗口存在，则该函数先生成标题为 Figure No.1 的新窗口，再在该窗口中操作；若已经有图形窗口存在，则在标号最高的图形窗口中进行操作。
- ezplot(f,[xmin xmax],fign)：在指定标号 fign 的窗口中、指定的范围[xmin xmax]内画出函数 f=f(x)的图形。
- ezplot(f,[xmin,xmax,ymin,ymax])：在平面矩形区域[xmin<x<xmax, ymin<y <ymax]上画出函数 f(x,y)=0 的图形。
- ezplot(x,y)：在默认的范围 $0<t<2\pi$ 内画参数形式函数 x=x(t)与 y=y(t)的图形。
- ezplot(x,y,[tmin,tmax])：在指定的范围[tmin < t < tmax]内画参数形式函数 x=x(t)与 y=y(t)的图形。
- ezplot(…,figure)：在由参量 figure 句柄指定的图形窗口中画函数图形。

【功能介绍】 绘制符号函数的图形。

【实例 4.33】 绘制符号方程 $2x^4-y^9=0$ 的曲线。

```
>>syms x y
>>ezplot(2*x^4-y^9)
```

这个隐函数的图形如图 4.1 所示。

【实例讲解】 从上面的例子中可以看出，使用 MATLAB 可以很便利地绘制隐函数的图形。

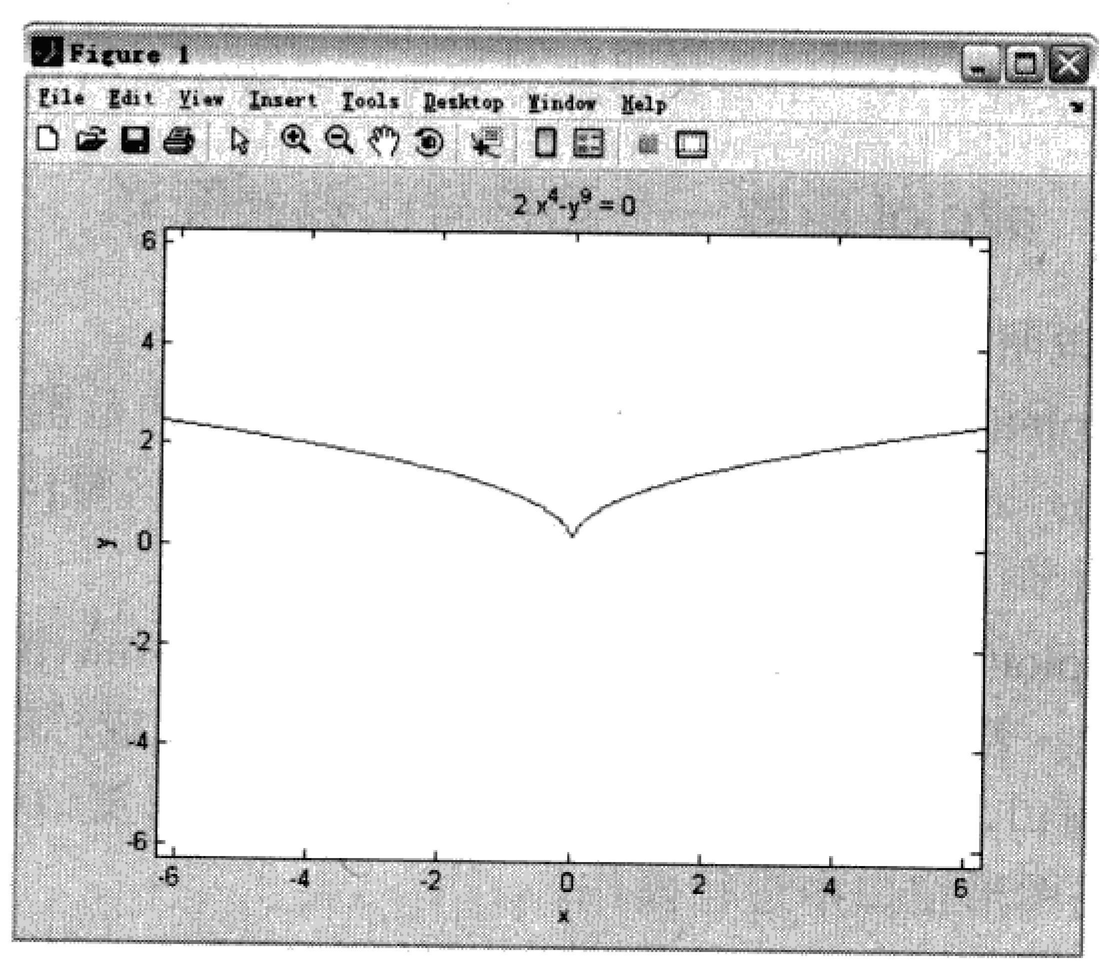


图 4.1　隐函数图形绘制

4.3.2　ezplot3 函数——三维曲线图

【语法说明】

- ezplot3(x,y,z)：在默认的范围 0<t<2π 内画空间参数形式的曲线 x=x(t)、y=y(t)与 z=z(t)的图形。
- ezplot3(x,y,z,[tmin,tmax])：在指定的范围 tmin < t < tmax 内画空间参数形式的曲线 x=x(t)、y=y(t)与 z=z(t)的图形。
- ezplot3(…,'animate')：以动画形式画出空间三维曲线。

【功能介绍】 绘制符号函数的三维曲线图。

【实例 4.34】 绘制符号函数的三维曲线图。

```
>>syms t;
>>ezplot3(t*sin(t), t*cos(t), t,[0,20*pi])
```

得到的结果如图 4.2 所示。

【实例讲解】 读者可以自行设置三维曲线图的视角，来查看不同的结果。

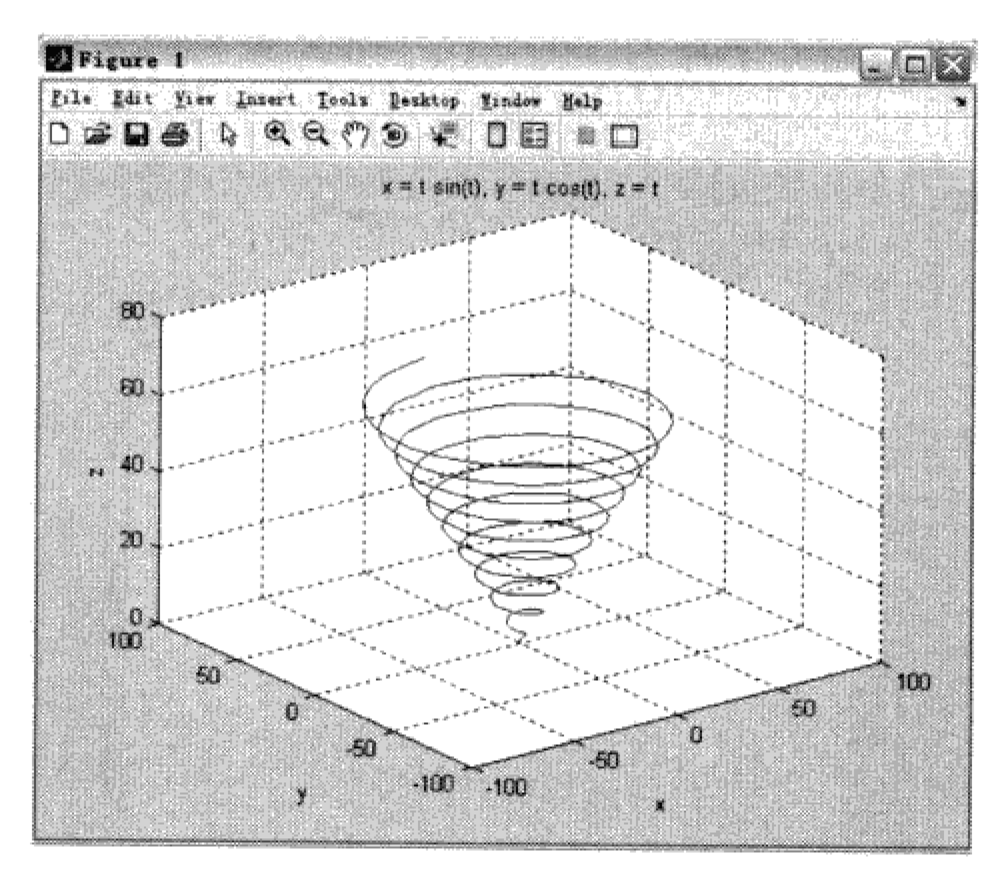


图 4.2　三维曲线图

4.3.3　ezcontour 函数——画符号函数的等高线图

【语法说明】

- ezcontour(f)：画出二元符号函数 $f=f(x,y)$的等高线图。函数 f 将被显示于默认的平面区域[-2π<x<2π，-2π<y<2π]内。系统将根据函数变动的激烈程度自动选择相应的计算栅格。若函数 f 在某些栅格点上没有定义，则这些点将不显示。
- ezcontour(f,domain)：在指定的定义域 domain 内画出二元函数 $f(x,y)$，参量 domain 可以是四维向量[xmin,xmax,ymin,ymax]或二维向量[min,max]（其中显示区域为：min<x<max，min<y<max）。
- ezcontour(…,n)：用指定 n×n 个栅格点（对定义域的一种划分），在默认（若没有指定）的区域内画出函数 f 的图形。n 的默认值为 60。

【功能介绍】 画符号函数的等高线图。

【实例 4.35】 画出下面函数的等高线图。

```
>>syms x y
>>f = (1-x)^2*exp(-(x^2)-(y+1)^2)-5*(x/5-x^3-y^5)*sin(-x^2-y^2)-1/3*exp(-(x+1)^2-y^2);
>>ezcontour(f,[-3,3],49)
```

得到的图形如图 4.3 所示。

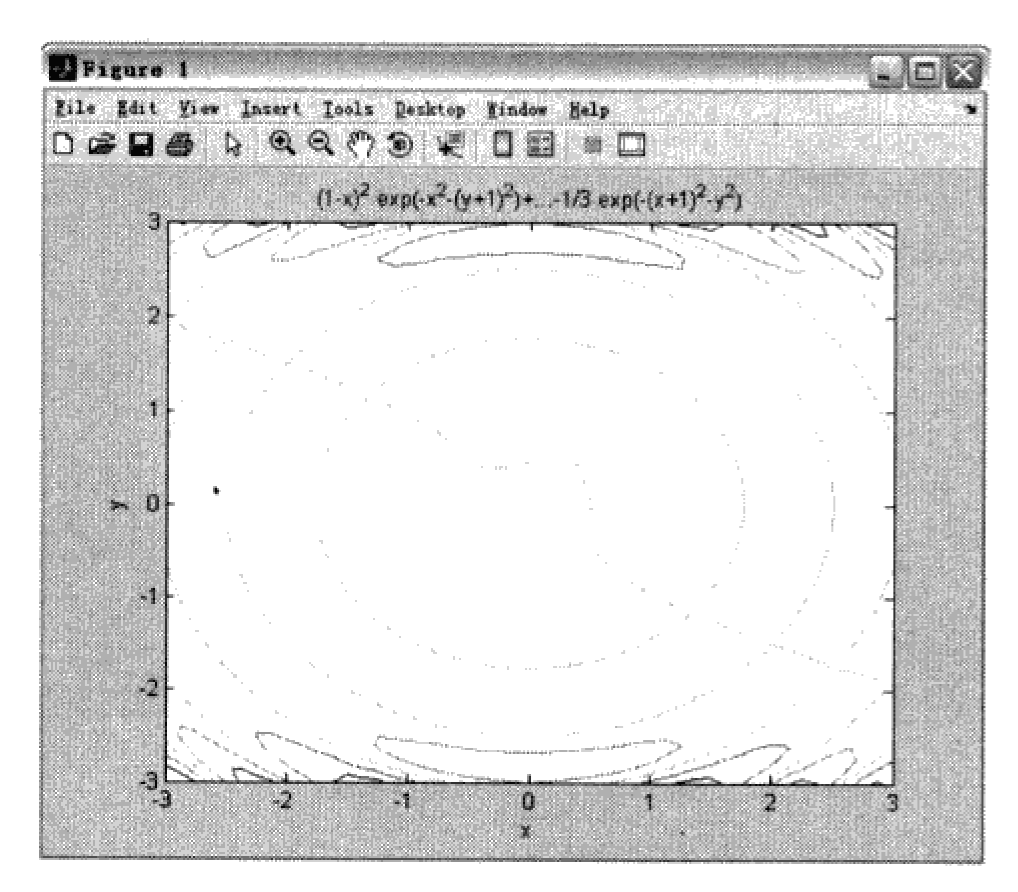


图 4.3　等高线绘制

【实例讲解】 等高线可以认为是用多个平面去截取二元函数的三维图形，得到的图形。

4.3.4 ezcontourf 函数——用不同颜色填充的等高线图

【语法说明】

- ezcontourf(f)：画出二元符号函数 f=f(x,y)的等高线图，且在不同的等高线之间自动用不同的颜色进行填充。函数 f 将被显示于默认的平面区域[-2π<x<2π,-2π<y<2π]内。系统将根据函数变动激烈程度自动选择相应的计算栅格。若函数 f 在某些栅格点上没有定义，这些点将不显示。
- ezcontourf(f,domain)：在指定的定义域 domain 内画出二元函数 f(x,y)的等高线图，且在不同的等高线之间自动用不同的颜色进行填充。定义域 domain 可以是四维向量[xmin,xmax,ymin,ymax]或二维向量[min,max]（其中显示区域为：min<x<max，min<y<max）。
- ezcontourf(…,n)：用指定 n×n 个栅格点（对定义域的一种

划分)，在默认（若没有指定）的区域内画出函数 f 的等高线图，且在不同的等高线之间自动用不同的颜色进行填充。n 的默认值为 60。

【功能介绍】 用不同颜色填充等高线图形。

【实例 4.36】 绘制等高线图，并对其着色。

```
>>syms x y
>>f = (1-x)^2*exp(-(x^2)-(y+1)^2)-5*(x/5-x^3-y^5)*sin
(-x^2-y^2)-1/3*exp(-(x+1)^2-y^2);
>>ezcontourf(f,[-3,3],64)
```

得到的图形如图 4.4 所示。

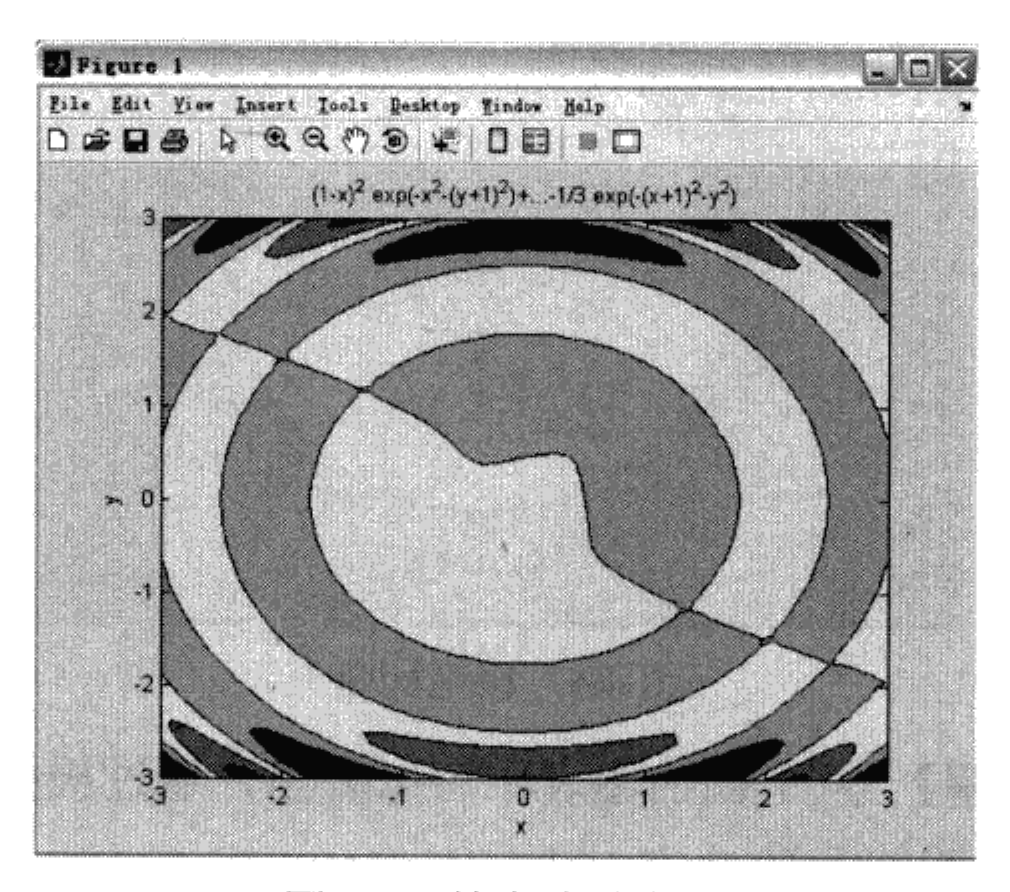


图 4.4　等高线填充图

【实例讲解】 根据等高线的定义，在上面的图形中，颜色相同的区域，二元函数的数值都相等。

4.3.5　ezpolar 函数——画极坐标图形

【语法说明】

- ezpolar(f)：在默认的范围 0<theta<2π 内画极坐标函数 rho=f(theta)的图形，且将函数关系式显示于图形下方。
- ezpolar(f,[a,b])：在指定的范围 a<theta<b 内画极坐标函数 rho=f(theta)的图形，且将函数关系式显示于图形下方。

【功能介绍】 绘制极坐标函数图形。

【实例 4.37】 绘制一个极坐标图形。

```
>>syms t
>>ezpolar(1+cos(5*t))
```

绘制了一个 1+cos(5×*t*)的极坐标图，如图 4.5 所示。

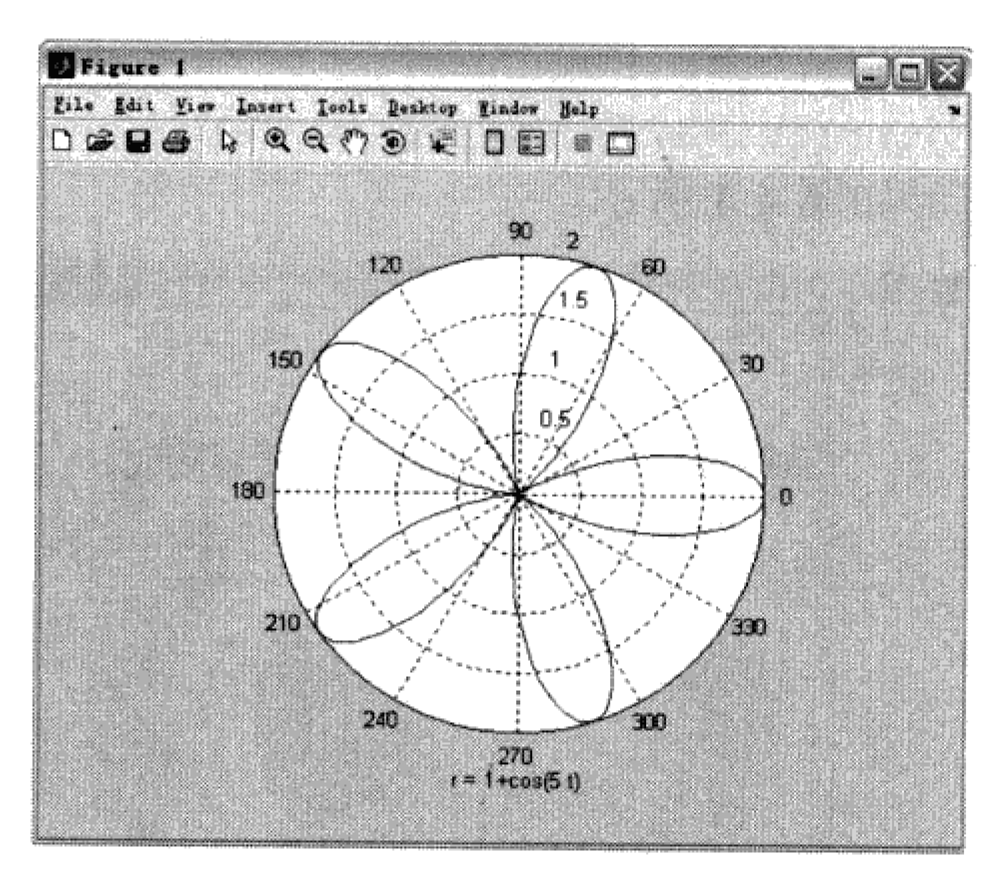


图 4.5 极坐标图

【实例讲解】 从上面的图表中可以看出，通过极坐标图可以很便利地查看函数的特性。

4.3.6 ezmesh 函数——符号函数的三维网格图

【语法说明】

- ezmesh(f)：画出二元数学符号函数 f=f(x,y)的网格图。函数 f 将显示于默认的平面区域[-2π<x<2π，-2π<y<2π]内。系统将根据函数变动的激烈程度自动选择相应的计算栅格。若函数 f 在某些栅格点上没有定义，则这些点将不显示。

- ezmesh(f,domain)：在指定的定义域 domain 内画出二元函数 f(x,y)的网格图，定义域 domain 可以是四维向量[xmin,xmax, ymin,ymax]或二维向量[min,max]（其中显示区域为：min<x<max，min<y<max）。

- ezmesh(x,y,z)：在默认的矩形定义域范围[-2π<s<2π,-2π<t<2π]内画参数形式函数 x=x(s,t)、y=y(s,t)、z=z(s,t)的二元函数 z=f(x,y)的网格图。
- ezmesh(x,y,z,[smin,smax,tmin,tmax])：在指定的矩形定义域范围[smin<s<smax， min<t<tmax]内画参数形式函数 x=x(s,t)、y=y(s,t)、z=z(s,t)的二元函数 z=f(x,y)的网格图。
- ezmesh(x,y,z,[min,max])：用指定的矩形定义域[min<x<max, min<y<max]画出函数 z=f(x,y)的网格图。
- ezmesh(f,…,n)：用指定 n×n 个栅格点，在默认（若没有指定）的区域内画出函数 f 的图形。n 的默认值为 60。
- ezmesh(…,'circ')：在一圆形区域（圆心位于定义域在中心）的范围内画出函数 f 的网格图形。

【功能介绍】 绘制符号函数的三维网格图。

【实例 4.38】 绘制符号函数 $-x\sin(x^2+y^2)$ 的三维网格图。

```
>>syms x y
>>ezmesh(x*sin(-x^2-y^2),40,'circ')
>>colormap [0 0 1]
```

得到的网格图如图 4.6 所示。

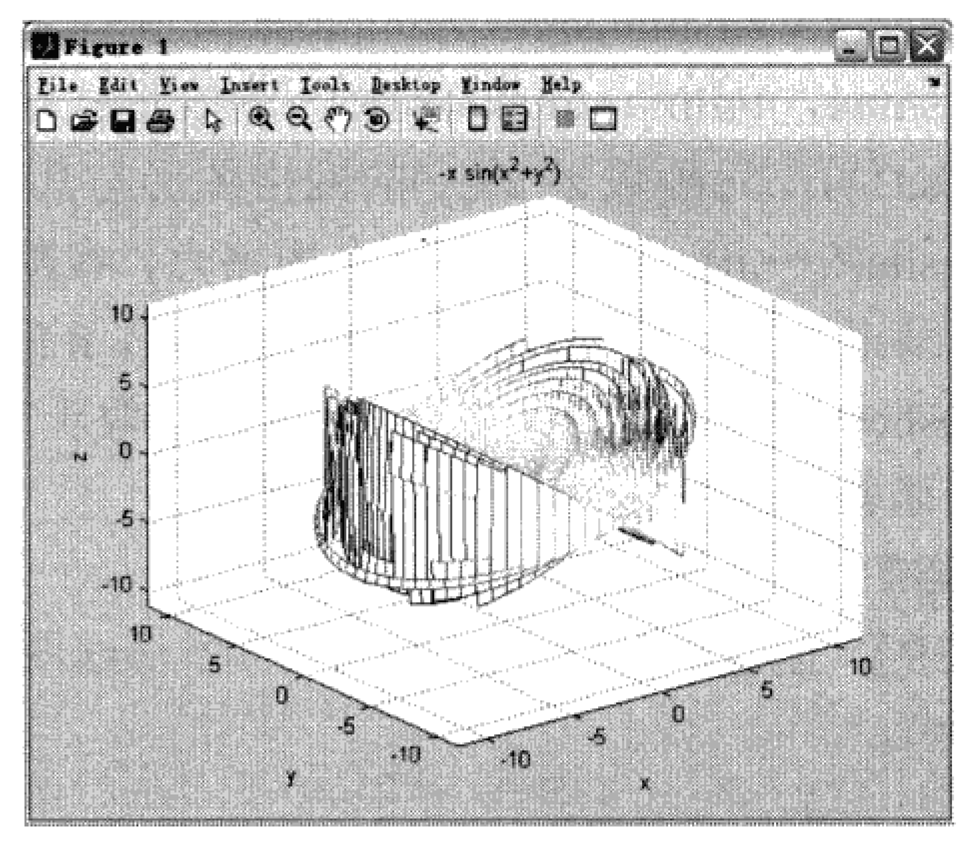


图 4.6 符号函数的三维网格图

【实例讲解】 在本例中，使用 colormap 函数来设置图形的配色方案。

4.3.7 ezmeshc 函数——同时画曲面网格图与等高线图

【语法说明】

- ezmeshc(f)：画出二元数学符号函数 z=f(x,y)的网格图形，同时在 xy 平面上显示其等高线图。函数 f 将被显示于默认的平面区域$[-2\pi<x<2\pi,-2\pi<y<2\pi]$内。系统将根据函数变动的激烈程度自动选择相应的计算栅格。若函数 f 在某些栅格点上没有定义，则这些点将不显示。
- ezmeshc(f,domain)：在指定的定义域 domain 内画出二元函数 f(x,y)的网格图及其等高线图，domain 可以是四维向量[xmin,xmax,ymin,ymax]或二维向量[min,max](其中显示区域为：$min<x<max$，$min<y<max$)。
- ezmeshc(x,y,z)：在默认的矩形定义域范围$[-2\pi<s<2\pi,-2\pi<t<2\pi]$内画出参数形式函数 x=x(s,t)、y=y(s,t)、z=z(s,t)的二元函数 z=f(x,y)的网格图形与其等高线图。
- ezmeshc(x,y,z,[smin,smax,tmin,tmax])：在指定的矩形定义域范围$[smin<s<smax，tmin<t<tmax]$内画出参数形式函数 x=x(s,t)、y=y(s,t)、z=z(s,t)的二元函数 z=f(x,y)的网格图形与其等高线图。
- ezmeshc(f,…,n)：用指定 n×n 个栅格点，在默认（若没有指定）的区域内画出函数 f 的网格图形与等高线图。n 的默认值为 60。
- ezmeshc(…,'circ')：在一圆形区域（圆心位于定义域在中心）的范围内画出函数 f 的网格图形及其等高线图。

【功能介绍】 绘制等高线图及三维网格图。

【实例 4.39】 创建函数表达式，然后绘制等高线图和三维网格图。

```
>>syms x y
>>ezmeshc(x*y/(1 + x^2 + y^2),[-5,5,-2*pi,2*pi],35)
```

等高线图和三维网格图显示在同一个图形中，如图 4.7 所示。

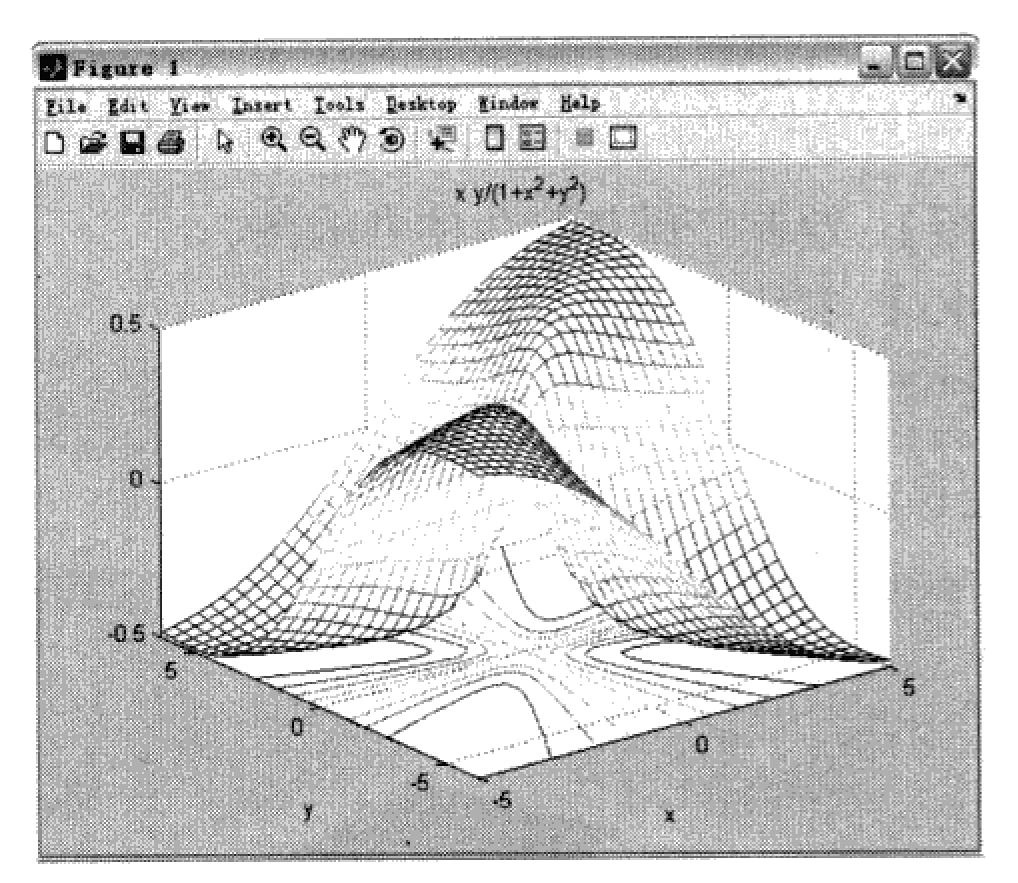


图 4.7　网格等高线图

【实例讲解】 用户可以重新设置等高线的间距，重新绘制图形。

4.3.8　ezsurf 函数——三维带颜色的曲面图

【语法说明】

- ezsurf(f)：画出二元数学符号函数 $z=f(x,y)$的曲面图形。函数 f 将显示于默认的平面区域[$-2\pi<x<2\pi$，$-2\pi<y<2\pi$]内。系统将根据函数的变动程度自动选择相应的计算栅格。若函数 f 在栅格点上没有定义，则这些点将不显示。
- ezsurf(f,domain)：在指定的定义域 domain 内画出二元函数 $f(x,y)$的曲面图形，domain 可以是四维向量[xmin,xmax,ymin,ymax]，或者是二维向量[min,max] (其中有 $min<x<max$，$min<y<max$)。
- ezsurf(x,y,z)：在默认的矩形定义域范围$-2\pi<s<2\pi$，$-2\pi<t<2\pi$ 内画出参数形式函数 $x=x(s,t)$、$y=y(s,t)$与 $z=z(s,t)$的曲面图形。
- ezsurf(x,y,z,[smin,smax,tmin,tmax])或 ezsurf(x,y,z,[min,max])：用指定的定义域画出参数形式的曲面图形。
- ezsurf(…,n)：用指定 n×n 个栅格点，在默认（若没有指定）的区域内画出函数 f 的图形，n 的默认值为 60。

- ezsurf(…,'circ')：在一圆形中心位于定义域范围内画出函数 f 的曲面图形。

【功能介绍】 绘制三维带颜色的曲线图。

【实例 4.40】 绘制三维带颜色的曲线图。

```
>>syms x y
>>ezsurf(real(atan(x+i*y)))
```

得到的图形如图 4.8 所示。

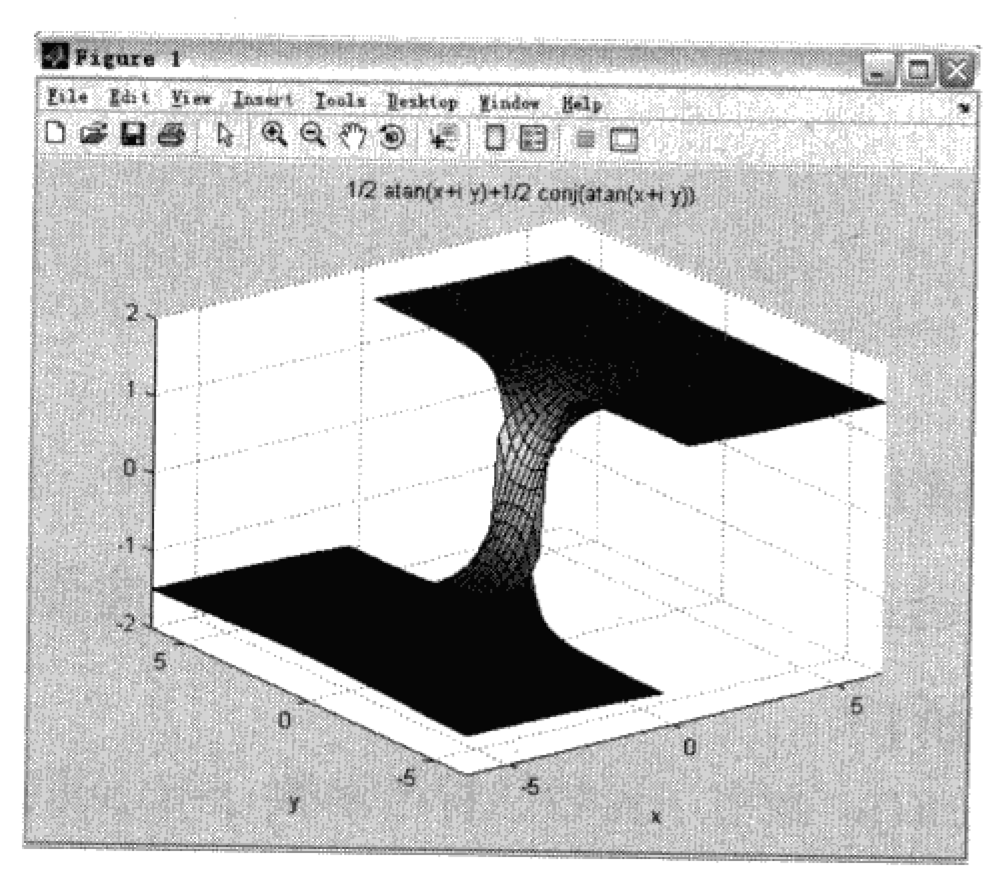


图 4.8　三维颜色曲线图

【实例讲解】 从上面的结果中可以看出，ezsurf 函数会自动为图形中的数据添加对应的颜色。用户可以根据需要设置不同的颜色参数。

4.3.9 ezsurfc 函数——同时画出曲面图与等高线图

【语法说明】

- ezsurfc(f)：画出二元数学符号函数 z=f(x,y)的曲面图形与其等高线图。函数 f 将显示于默认的平面区域[$-2\pi<x<2\pi$，$-2\pi<y<2\pi$]内。系统将根据函数的变动程度自动选择相应的计算栅格。若函数 f 在栅格点上没有定义，则这些点将不显示。
- ezsurfc(f,domain)：在指定的定义域 domain 内画出二元函数

f(x,y)的曲面图形及其等高线图，domain 可以是四维向量[xmin, xmax, ymin,ymax]或二维向量[min,max](其中有 $min<x<max$，$min<y<max$)。

- ezsurfc(x,y,z)：在默认的矩形定义域范围 $-2\pi<s<2\pi$，$-2\pi<t<2\pi$ 内画出参数形式函数 x=x(s,t)、y=y(s,t)与 z=z(s,t)的曲面图形与等高线图。
- ezsurfc(x,y,z,[smin,smax,tmin,tmax]) 或 ezsurfc(x,y,z,[min, max])：用指定的定义域画出参数形式的曲面图形与等高线图。
- ezsurfc(…,n)：用指定 n×n 个栅格点，在默认（若没有指定）的区域内画出函数 f 的曲面图形与等高线图，n 的默认值为 60。
- ezsurfc(…,'circ')：在一圆形中心位于定义域的中心范围内画出函数 f 的曲面图形与等高线图

【功能介绍】 绘制曲线图与等高线。

【实例 4.41】 绘制曲面图和等高线。

```
>>syms x y
>>ezsurfc(x*y/(1+x^2 + y^2),[-5,5,-2*pi,2*pi],35,'circ')
```

得到的三维曲面等高线图如图 4.9 所示。

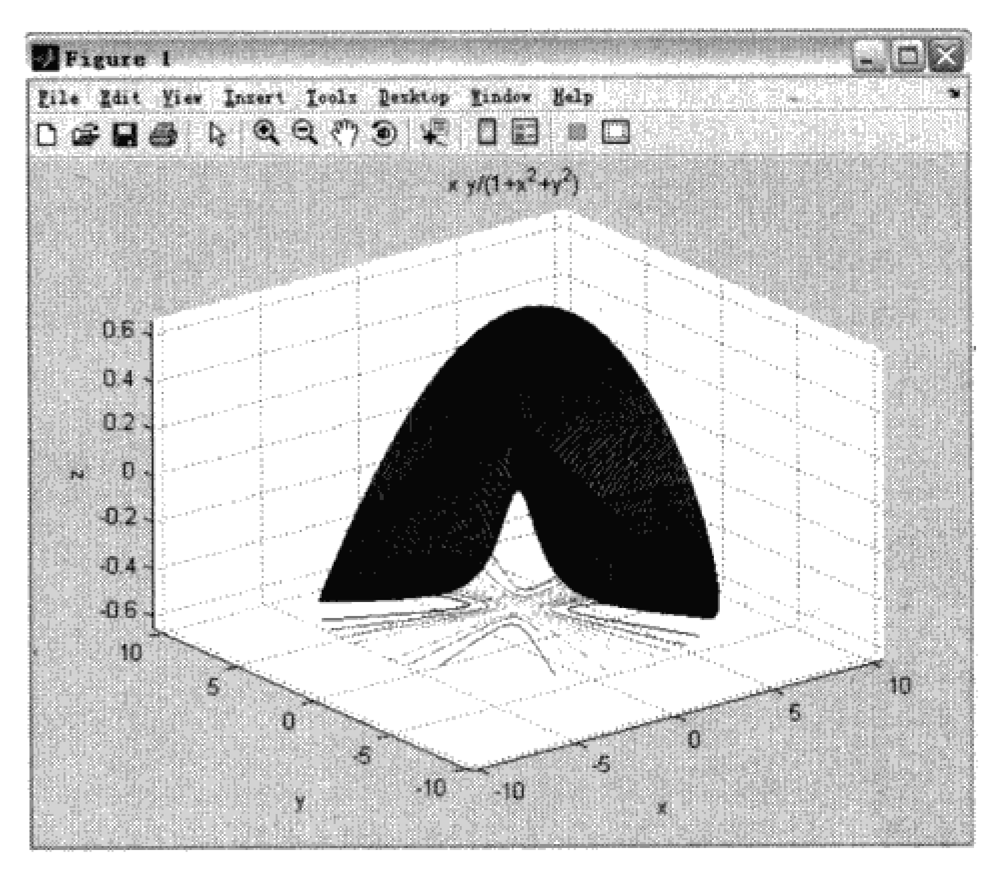


图 4.9　三维曲面等高线图

【实例讲解】 在上面的图形中，三维曲面的颜色代表的是数值

的大小。

4.4 积分变换

积分变换在电子电路、自动控制原理等学科中应用极为广泛，通常借助 MATLAB 的强大功能计算积分变换。下面就 Fourier 积分变换、Laplace 变换和 z 变换这 3 种最普遍使用的积分变换形式进行讲解。

4.4.1 fourier 函数——Fourier 积分变换

【语法说明】 F = fourier(f)：对符号单值函数 f 中的默认变量 x（由函数 findsym 确定）计算 Fourier 变换形式。

【功能介绍】 求积分变换。

【实例 4.42】 对下面几个函数求积分变换。

```
>>syms x w u
>>f 1= sin(x)*exp(-x^2);
>> F1 = fourier(f1)
>>f2 = log(abs(w));
>> F2 = fourier(f2)
>>f3 = x*exp(-abs(x));
>> F3 = fourier(f3,u)
```

计算结果为：

```
F1 =
    -1/2*i*pi^(1/2)*exp(-1/4*(w-1)^2)+1/2*i*pi^(1/2)
*exp(-1/4*(w+1)^2)
F2 =
    fourier(log(abs(w)),w,t)
F3 =
   -4*i/(1+u^2)^2*u
```

【实例讲解】 如果函数没有指定积分变换后的系数，那么系统将变换后的函数自变量默认为“w”；如果指定了自变量，那么结果就为以所指定参数为自变量的函数，如上例中最后一个函数所示。

4.4.2　ifourier 函数——逆 Fourier 积分变换

【语法说明】

- f = ifourier(F,u)：使函数 f 为变量 u（u 为标量符号对象）的函数，$f(u)=\frac{1}{2\pi}\int_{-\infty}^{\infty}F(w)e^{iwu}dw$。
- f = ifourier(F,v,u)：使 F 为变量 v 的函数，f 为变量 u 的函数，$f(u)=\frac{1}{2\pi}\int_{-\infty}^{\infty}F(v)e^{iwu}dv$。

【功能介绍】 求 Fourier 积分逆变换。

【实例 4.43】 分别对下面的几个符号函数求 Fourier 积分逆变换。

```
>>syms w v x t
>>syms a real
>>f = sqrt(exp(-w^2/(4*a^2)));
>>IF1 = ifourier(f)
>>g = exp(-abs(x));
>>IF2 = ifourier(g)
>>h = sinh(-abs(w)) - 1;
>>IF3 = simple(ifourier(h,t))
>>syms w real
>>k = exp(-w^2*abs(v))*sin(v)/v;
>>IF4 = ifourier(k,v,t)
```

计算结果为：

```
IF1 =
    ifourier(exp(-1/4*w^2/a^2)^(1/2),w,x)
IF2 =
    1/(1+t^2)/pi
IF3 =
   -1/2*(pi*ifourier(exp(abs(w)),w,t)+pi*ifourier(exp
(abs(w)),w,t)*t^2-…  1+2*pi*Dirac(t))/(1+t^2)/pi
IF4 =
    1/2*(atan((t+1)/w^2)-atan((t-1)/w^2))/pi
```

【实例讲解】 读者可以将这个函数和前面的函数结合起来分析。

4.4.3 laplace 函数——Laplace 变换

【背景知识】 Laplace 变换的定义是这样的：$L(s)=\int_0^{\infty}F(t)\mathrm{e}^{-st}\mathrm{d}t$，laplace(F,t) 使函数 L 为变量 t（t 为标量符号自变量）的函数，$L(s)=\int_0^{\infty}\mathrm{F}(x)\mathrm{e}^{-tx}\mathrm{d}x$。变换形式：$F=F(s)\rightarrow L=L(t)$。

【语法说明】 L= laplace(F)：输出参量 L = L(s)为有缺省符号自变量 t 的标量符号对象 F 的 Laplace 变换。即：F = F(t) → L = L(s)。若 F = F(s)，则 fourier(F)返回变量为 t 的函数 L。

【功能介绍】 进行 Laplace 变换。

【实例 4.44】 分别对以下几个符号函数分别求变换，$f1=\sqrt{t}$，$f2=\dfrac{1}{\sqrt{t}}$，$f3=e^{-at}$。

```
>>syms x s t v
>>f1= sqrt(t);
>>L1 = laplace(f)
>>f2 = 1/sqrt(s);
>>L2 = laplace(f2)
>>f3 = exp(-a*t);
>>L3 = laplace(f3,x)
```

计算结果为：

```
L1 =
    1/(s-1/s^2)
L2 =
    (pi/t)^(1/2)
L3 =
    1/(x+a)
```

【实例讲解】 读者可以根据理论进行 Laplace 变换，然后进行对比。

4.4.4 ilaplace 函数——逆 Laplace 变换

【语法说明】

- F = ilaplace(L)：输出参量 F = F(t)为缺省变量 s 的标量符号

对象 L 的逆 Laplace 变换。F = F(w) → f = f(x)。若 L = L(t)，则 ifourier(L)返回变量为 x 的函数 F。即：F = F(x) → f = f(t)。逆 Laplace 变换定义为：$F(t)=\int_{c-i\infty}^{c+i\infty}L(s)e^{st}dt$。

- F = ilaplace(L,y)：使函数 F 为变量 y（y 为标量符号对象）的函数，$F(y)=\int_{c-i\infty}^{c+i\infty}L(y)e^{sy}ds$。
- F = ilaplace(L,y,x)：使 F 为变量 x 的函数，L 为变量 y 的函数，$F(x)=\int_{c-i\infty}^{c+i\infty}L(y)e^{xy}dy$。

【功能介绍】 求逆 Laplace 变换。

【实例 4.45】 创建符号表达式，然后进行 Laplace 变换。

```
>>syms a s t u v x
>>f = exp(x/s^2);
>>IL1 = ilaplace(f)
>>g = 1/(t-a)^2;
>>IL2 = ilaplace(g)
>>k = 1/(u^2-a^2);
>>IL3 = ilaplace(k,x)
>>y = s^3*v/(s^2+v^2);
>>IL4 = ilaplace(y,v,x)
```

计算结果为：

```
IL1 =
     ilaplace(exp(x/s^2),s,t)
IL2 =
     x*exp(a*x)
IL3 =
    1/(-a^2)^(1/2)*sin((-a^2)^(1/2)*x)
IL4 =
    s^3*cos((s^2)^(1/2)*x)
```

【实例讲解】 读者可以根据理论进行逆 Laplace 变换，然后进行对比。

4.4.5 ztrans 函数——求 z-变换

【背景知识】 Z 变换为信号处理中非常普遍常用的变换形式。

其输出参量 F 为变量 z 的函数，f = f(n) → F = F(z)。函数 f 的 z-变换定义为：$F(z)=\sum_{n=0}^{\infty}\frac{f(n)}{z^n}$。若函数 f = f (z)，则 ztrans(f)返回一变量为 w 的函数：f = f(z) → F = F(w)。

【语法说明】

- F=ztrans(f)：对缺省自变量为 n（就像由函数 findsym 确定的一样）的单值函数 f 计算 z-变换。
- F=ztrans(f,w)：用符号变量 w 代替缺省的 z 作为函数 F 的自变量 $F(w)=\sum_{n=0}^{\infty}\frac{f(n)}{w^n}$。
- F=ztrans(f,k,w)：对函数 f 中指定的符号变量 k 计算 z-变换：$F(w)=\sum_{n=0}^{\infty}\frac{f(k)}{w^n}$。

【功能介绍】 求 z 变换。

【实例 4.46】 创建符号表达式，并对其进行 z-变换。

```
>>syms a k w x n z
>>f1 = n^4;
>>ZF1 = ztrans(f)
>>f2 = a^z;
>>ZF2 = ztrans(g)
>>f3 = sin(a*n);
>>ZF3 = ztrans(f,w)
>>f4 = exp(k*n^2)*cos(k*n);
>>ZF4 = ztrans(f,k,x)
```

计算结果为：

```
ZF1 =
        z*(z^3+11*z^2+11*z+1)/(z-1)^5
ZF2 =
        w/a/(w/a-1)
ZF3 =
       -w*sin(a)/(-w^2+2*w*cos(a)-1)
ZF5 =
       (x/exp(n^2)-cos(n))*x/exp(n^2)/(x^2/exp(n^2)
^2-2*x/exp(n^2)*cos(n)+1)
```

【实例讲解】 对于比较复杂结果的情况，用户可以使用 simple 函数将其简化。

4.4.6　iztrans 函数——逆 z-变换

【语法说明】

- f = iztrans(F)：F = F(n) → f = f(k)。逆 z-变换定义为：$f(n)=\frac{1}{2\pi i}\oint_{|z|=R}F(z)z^{n-1}\mathrm{d}z$，n =1,2,3,…其中 R 为一正实数，它使函数 F(z)在圆域之外 |z|≥R 是解析的。
- f = iztrans(F,k)：使函数 f 为变量 k（k 为标量符号对象）的函数 f(k)：$f(k)=\frac{1}{2\pi i}\oint_{|z|=R}F(z)z^{k-1}\mathrm{d}z$，k=1,2,3,…
- f = iztrans(F,w,k)：使函数 F 为变量 w 的函数，f 为变量 k 的函数，$f(\mathrm{k})=\frac{1}{2\pi i}\oint_{|w|=R}F(w)w^{k-1}\mathrm{d}w$，k=1,2,3,…

【功能介绍】 求解逆 z-变换。

【实例 4.47】 创建符号表达式，然后对其进行逆 z-变换。

```
>>syms a n k x z
>>f1= 2*z/(z^2+2)^2;
>>IZ1 = iztrans(f1)
>>f2 = n/(n+1);
>>IZ2 = iztrans(f2)
>>f3 = z/sqrt(z-a);
>>IZ3 = iztrans(f3,k)
>>f4 = exp(z)/(x^2-2*x*exp(z));
>>IZ4 = iztrans(f4,x,k)
```

计算结果为：

```
    IZ1 =
         -1/8*sum(1/_alpha*(1/_alpha)^n,_alpha
    IZ2 =
         (-1)^k
    IZ3 =
         iztrans(z/(z-a)^(1/2),z,k)
```

```
        IZ4 =
                1/4*(-charfcn[0](k)-2*charfcn[1](k)*exp(z)
+2^k*exp(z)^k)/exp(z)
```

【实例讲解】 输出参量$f=f(n)$为有缺省变量 z 的单值符号函数 F 的逆 z-变换。即：$F=F(z)\ \rightarrow\ f=f(n)$。若 $F=F(n)$，则 iztrans(F)返回变量为 k 的函数$f(k)$。

4.5 其他符号运算函数

这一节将要讲解其他的一些经常使用的函数，如变精度算法、泰勒级数展开、雅可比矩阵、Maple 内核操作、符号表达式的其他语言表示等。

4.5.1 vpa 函数——可变精度算法计算

【语法说明】

- R = vpa(A)：用可变精度算法来计算 A 中的每一元素，使其成为有 d 位精确度的十进制数。其中 d 为函数 digits 设置的当前位数。R 中的每一元素为一符号表达式。
- R = vpa(A,d)或 R = vpa A d：用参量 d 指定的位数（而非函数 digits 设置的位数）来表示 A 中的每一元素。R 中的每一个元素为一个符号表达式。

【功能介绍】 用可变精度算法计算矩阵中的每一个元素。

【实例 4.48】 对几个简单的符号表达式用指定的精度来计算其结果。

```
>>digits(16)
>>a = vpa(sym(sin(pi/6)))
>>b = vpa(pi)
>>gold = vpa('(sqrt(5)-1)/2')
>>vpa pi 60
>>A = vpa(gallery(5),8)
>>B = vpa(hilb(3),5)
```

计算结果为：

```
a =
.8660254037844385
b =
3.141592653589793
gold =
.6180339887498950
ans =
3.14159265358979323846264338327950288419716939937510
582097494
A =
    [     -9.,      11.,     -21.,      63.,    -252.]
    [     70.,     -69.,     141.,    -421.,    1684.]
    [   -575.,     575.,   -1149.,    3451.,  -13801.]
    [   3891.,   -3891.,    7782.,  -23345.,   93365.]
    [   1024.,   -1024.,    2048.,   -6144.,   24572.]
B =
    [     1., .50000, .33333]
    [ .50000, .33333, .25000]
    [ .33333, .25000, .20000]
```

【实例讲解】 vpa 函数中指定了用来计算的精度，那么，得到的结果小数点后保留的就是该精度值的位数。

4.5.2　subs 函数——在一符号表达式或矩阵中进行符号替换

【语法说明】

- R = subs(S)：用从调用的函数中获得的变量值，或 MATLAB 的工作空间中存在的变量值，替换表达式 S 中所有出现的相同的变量，同时自动进行化简计算；若是数值表达式，则计算出结果。
- R = subs(S,old,new)：用新值 new 替换表达式 s 中的旧值 old，参量 old 是一符号变量或代表一变量名的字符串，new 是一符号（数值）变量或表达式。

【功能介绍】 在符号表达式或矩阵中进行符号替换。

4.5.3 taylor 函数——符号函数的 Taylor 级数展开式

【语法说明】

- r = taylor(f,n,v)：返回符号表达式 f 中的、指定的符号自变量 v（若表达式 f 中有多个变量时）的 n–1 阶的 Maclaurin 多项式（即在零点附近 v=0）近似式，其中 v 可以是字符串或符号变量。
- r = taylor(f)：返回符号表达式 f 中的、符号变量 v 的 6 阶的 Maclaurin 多项式（即在零点附近 v=0）近似式，其中 v=findsym(f)。
- r = taylor(f,n,v,a)：返回符号表达式 f 中的、指定的符号自变量 v 的 n–1 阶的 Taylor 级数（在指定的 a 点附近 v=a）的展开式。其中 a 可以是一数值、符号、代表一数字值的字符串或未知变量。解析函数 $f(x)$在点 x=a 的 Taylor 级数定义为：$f(x)=\sum_{n=0}^{\infty}\frac{f^{(n)}(a)}{n!}(x-a)^n$

【功能介绍】 求 Taylor 级数展开式。

【实例 4.49】 求下面函数 $f=\sin\left(x+\frac{\pi}{3}\right)$的 Taylor 级数展开式。

```
>>syms x y a pi m m1 m2
>>f = sin(x+pi/3);
>>T1 = taylor(f)
>>T2 = taylor(f,9)
>>T3 = taylor(f,a)
>>T4 = taylor(f,m1,m2)
>>T5 = taylor(f,m,a)
>>T6 = taylor(f,y)
>>T7 = taylor(f,y,m)   % 也可以用 taylor(f,m,y)替换
>>T8 = taylor(f,m,y,a)
>>T9 = taylor(f,y,a)
```

计算结果为：

```
T1 =
1/2*3^(1/2)+1/2*x-1/4*3^(1/2)*x^2-1/12*x^3+1/48*3^(1/2)
*x^4+1/240*x^5
T2 =
1/2*3^(1/2)+1/2*x-1/4*3^(1/2)*x^2-1/12*x^3+1/48*3^(1
```

```
/2)*x^4+1/240*x^5-1/1440*3^(1/2)* x^6-1/10080*x^7+1/80640*3^
(1/2)*x^8
    T3 =
    sin(a+1/3*pi)+cos(a+1/3*pi)*(x-a)-1/2*sin(a+1/3*pi)*
(x-a)^2-1/6*cos(a+1/3*pi)*    (x-a)^3+1/24*sin(a+1/3*pi)*
(x-a)^4+1/120*cos(a+1/3*pi)*(x-a)^5
    T4 =
    sin(m2+1/3*pi)+cos(m2+1/3*pi)*(x-m2)-1/2*sin(m2+1/3*
pi)*(x-m2)^2-1/6*  cos(m2+1/3*pi)*(x-m2)^3+1/24*sin(m2+1/
3*pi)*(x-m2)^4+1/120*
    cos(m2+1/3*pi)*(x-m2)^5
    T5 =
    sin(a+1/3*pi)+cos(a+1/3*pi)*(x-a)-1/2*sin(a+1/3*pi)*
(x-a)^2-1/6*cos(a+1/3*pi)*  (x-a)^3+1/24*sin(a+1/3*pi)*(x-a)
^4+1/120*cos(a+1/3*pi)*(x-a)^5
    T6 =
    sin(y+1/3*pi)+cos(y+1/3*pi)*(x-y)-1/2*sin(y+1/3*pi)*
(x-y)^2-1/6*cos(y+1/3*pi)  *(x-y)^3+1/24*sin(y+1/3*pi)*(x-y)
^4+1/120*cos(y+1/3*pi)*(x-y)^5
    T7 =
    sin(m+1/3*pi)+cos(m+1/3*pi)*(x-m)-1/2*sin(m+1/3*pi)*
(x-m)^2-1/6*cos(m+1/3*pi)  *(x-m)^3+1/24*sin(m+1/3*pi)*(x-m)
^4+1/120*cos(m+1/3*pi)*(x-m)^5
    T8 =
    sin(a+1/3*pi)+cos(a+1/3*pi)*(x-a)-1/2*sin(a+1/3*pi)*
(x-a)^2-1/6*cos(a+1/3*pi)*  (x-a)^3+1/24*sin(a+1/3*pi)*(x-a)
^4+1/120*cos(a+1/3*pi)*(x-a)^5
    T9 =
    sin(a+1/3*pi)+cos(a+1/3*pi)*(x-a)-1/2*sin(a+1/3*pi)*
(x-a)^2-1/6*cos(a+1/3*pi)*  (x-a)^3+1/24*sin(a+1/3*pi)*(x-a)
^4+1/120*cos(a+1/3*pi)*(x-a)^5
```

【实例讲解】 T2 指定了展开的阶数，为 9−1=8 阶，T3 指定了在 a 点展开。

4.5.4 jacobian 函数——求 Jacobian 矩阵

【语法说明】 R = jacobian(w,v)：计算 w 对 v 的 Jacobian 矩阵。其中 w 为符号单值函数表达式或符号列向量，v 为一符号行向量。输出参量 R=（r_{ij}）的元素 r_{ij} 为：$r_{ij}=\dfrac{\partial w(i)}{\partial v(j)}$，i=1,2,…,length(w)，j=1,

2,…,length(v)。

【功能介绍】 求解 Jacobian 矩阵。

【实例 4.50】 求解 Jacobian 矩阵。

```
>>syms x y z a b c
>>w = [x*y*z; y; x+z];
>>v = [x,y,z];
>>R = jacobian(w,v)
>>P = jacobian(x+u, v)
```

计算结果为：

```
R =
    [ y*z,  x*z,  x*y]
    [   0,    1,    0]
    [   1,    0,    1]
P =
[ 1, 0, 0]
```

【实例讲解】 R 为求得的 w 对 v 的 Jacobian 矩阵，P 为 u+v 对 v 的 Jacobian 矩阵。

4.5.5 jordan 函数——Jordan 标准形

【语法说明】

- J = jordan(A)：计算矩阵 A 的 Jordan 标准形。其中 A 为一确切已知的符号或数值矩阵。即它的元素必须是整数或小整数的比值。任何的矩阵输入误差将导致不同的 Jordan 标准形。即 Jordan 标准形对数据是敏感的。
- [V,J] = jordan(A)：返回 Jordan 标准形矩阵 J 与相似变换矩阵 V，其中 V 的列向量为矩阵 A 的广义特征向量。它们满足：V\A*V=J。

【功能介绍】 求矩阵的 Jordan 标准形。

【实例 4.51】 求解矩阵的 Jordan 标准形。

```
>>A = [1 -3 -2; -1  1 -1; 2 4 5]
>> [V,J] = jordan(A)
>>V = double(V);
>>Test = all(all(V\A*V == J))
```

计算结果为：

```
V =
    -1    -1     1
     0    -1     0
     1     2     0
J =
     3     0     0
     0     2     1
     0     0     2
Test = 1
```

【实例讲解】 Jordan 标准形是线性代数中的一个重要理论部分，用户可以查阅相应的线性代数书籍。

4.5.6　rsums 函数——交互式计算 Riemann

【语法说明】 rsums(f)：交互式地通过 Riemann 计算函数 $f(x)$ 的积分。rsums(f)显示函数 f 的图形。用户可以通过拖动图形下方的滑块来调整 Riemann 的项数，有效的项数为 2～128。

【功能介绍】 通过 Riemann 交互式地计算函数 $f(x)$的积分。

【实例 4.52】 显示交互式计算。

```
>>rsums sin(-3*sinx^2)
```

得到的图形如图 4.10 所示。

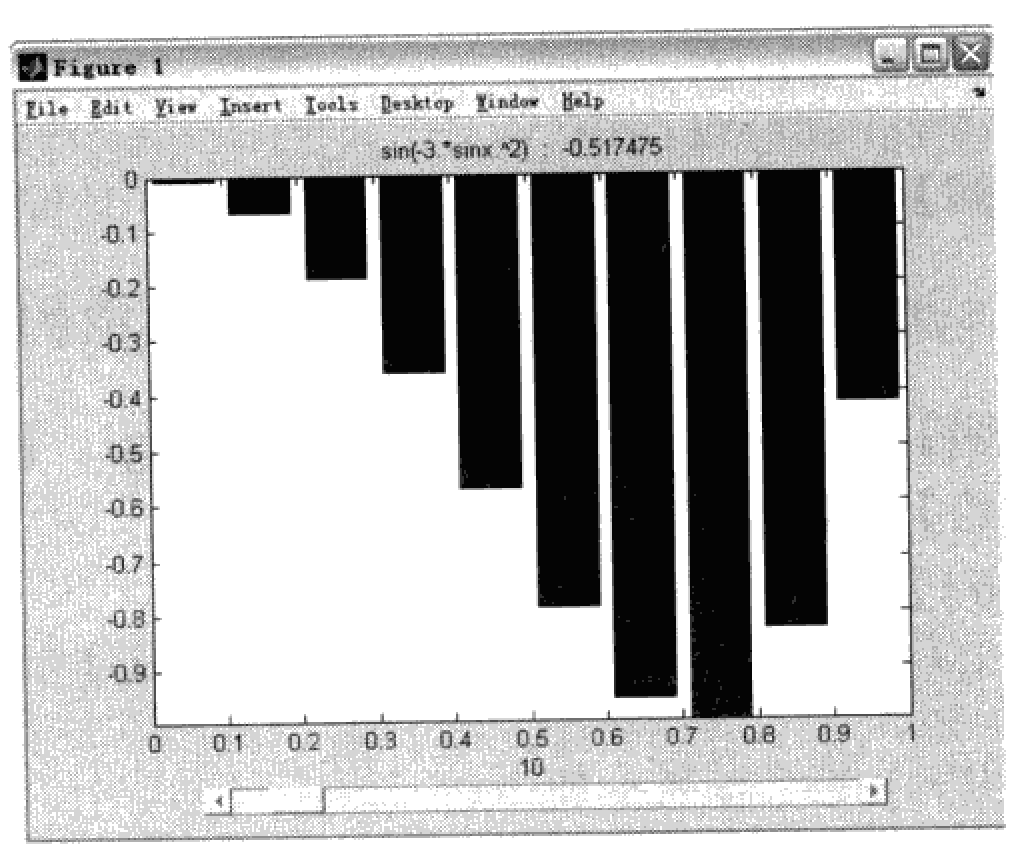


图 4.10　函数的 Riemann

【实例讲解】 读者可以在上面的对话框中调整按钮，查看动态效果。

4.5.7 latex 函数——符号表达式的 LaTex 的表示式

【语法说明】 latex(S)：返回符号表达式 S 的 LaTex 格式的表示式。该格式可以使表达式 S 在图形窗口中进行显示。

【功能介绍】 求解符号表达式的 LaTex 格式的表示式。

【实例 4.53】 求解符号表达式的 LaTex 的表达形式。

```
>>syms x
>>f = taylor(sin(1+x));
>>Lat1 = latex(f)
>>M = sym(magic(3));
>>Lat2 = latex(M)
```

计算结果为：

```
Lat1 =
\sin(1)+\cos(1)\mbox {{\tt 'x~'}}-1/2\,\sin(1){\mbox
{{\tt 'x~'}}}^{2}-1/6\,\cos(1){\mbox {{\tt 'x~'}}}^{3}
+1/24\,\sin(1){\mbox {{\tt 'x~'}}}^{4}+{\frac {1}{120}}\,
\cos(1){\mbox {{\tt 'x~'}}}^{5}
Lat2 =
\left [\begin {array}{ccc} 8&1&6\\\noalign{\medskip}
3&5&7\\\noalign{\medskip}4…
&9&2\end {array}\right ]
```

【实例讲解】 Latex 是一个通用的编排软件；功能强大。

4.5.8 syms 函数——创建多个符号对象的快捷函数

【语法说明】 syms arg1 arg2 …：定义 arg1、arg2 为符号。

【功能介绍】 创建快捷函数。

【实例 4.54】 创建符号对象。

```
>>syms x beta real  %符号对象生成,执行下面操作:
>>whos
```

工作空间中详细信息显示如下：

```
      Name          Size          Bytes  Class
       beta         1x1            132   sym object
```

```
    x          1x1            126  sym object
   Grand total is 7 elements using 258 bytes
y = x + i*beta; clear x; y
```

【实例讲解】 从上面的结果中可以看出，使用 syms 可以一次性创建多个对象。

4.5.9 maple 函数——调用 Maple 内核

【语法说明】

- r = maple('statement')：将参数函数 statement 传递给 Maple 内核，且返回计算结果。
- r = maple('function',arg1,arg2,⋯)：该函数接受任何的带引号的函数名 function 与相关的输入参量 arg1,arg2,⋯在必要时，要将输入参量转换成符号表达式。
- [r, status] = maple(⋯)：有条件地返回警告/错误信息。当语句能顺利执行，则 r 为计算结果，status 为 0；若语句不能通过执行，r 为相应的警告/错误信息，而 status 为一正整数。
- maple('traceon') 、maple traceon、maple trace on：将显示所有的后面的 Maple 语句与其相应的结果于屏幕上。
- maple('traceoff') 、maple traceoff、maple trace off：将关闭上面的操作特性。

【功能介绍】 调用 Maple 内核。

【实例 4.55】 调用 Maple 内核，进行数学计算。

```
>>Pi = maple('evalf(Pi,100)')
>>syms x
>>v = [x^2-1;x^2-4]
>>maple traceon
>>w = factor(v)
```

计算结果为：

```
Pi =
3.1415926535897932384626433832795028841971693993751O
58209749445923078164…
                    06286208998628034825342117068
v =
```

```
     [ x^2-1]
     [ x^2-4]
 statement:
    map(ifactor,array([[x^2-1],[x^2-4]]));
 result:
    Error, (in ifactor) invalid arguments
 statement:
    map(factor,array([[x^2-1],[x^2-4]]));
 result:
    matrix([[(x-1)*(x+1)], [(x-2)*(x+2)]])
 w =
 [ (x-1)*(x+1)]
 [ (x-2)*(x+2)]
```

【实例讲解】 结果显示为调用内核计算的结果。

4.5.10 mfun 函数——Maple 数学函数的数值计算

【语法说明】 Y = mfun('function',par1,par2,par3,par4)：计算一指定的 Maple 软件中已知的数学函数 function 的数值。每一参量 par 为该函数相应的具体数值。用户可以输入满 4 个参量。最后指定的参量可以是矩阵，通常对应于 x。

【功能介绍】 计算已知的数学函数 function 的数值。

【实例 4.56】 计算数学函数 function 的数值。

```
>>M1 = mfun('dilog',5)
>>M2 = mfun('Psi',[3*i 0])
```

计算结果为：

```
M1 =
    -2.3699
M2 =
   1.1080 + 1.7375i      NaN
```

【实例讲解】 'dilog' 和 'Psi' 均是已知的数学函数 function，得到的结果是通过这两个函数计算得到的数值。

4.5.11 mhelp 函数——Maple 函数帮助

【语法说明】 mhelp topic、mhelp('topic')：返回 Maple 软件中指

定的 Maple 标题 topic 的在线帮助文档信息。

【功能介绍】 显示 Maple 函数帮助。

【实例 4.57】 显示 Maple 的帮助信息。

```
>> mhelp topic;
```

结果显示为：

```
    Display a Known Help Topic
    If you know the topic of the help page you want to read, 
you can access the help
    page directly from the worksheet.
       1. Type ?topic at the prompt.
       2. Press the Enter key or the Return key to display 
the help page for topic.
    See Also
      worksheet,help,fulltextsearch
      worksheet,help,topicsearch
      worksheet,help,history
     worksheet,reference,browse
      worksheet,reference,HelpGuide
    Reference Pages
      Help
```

【实例讲解】 MATLAB 集成了 Maple 中的大部分功能，用户可以直接在 MATLAB 中查看 Maple 的多种信息。

4.5.12 sym2poly 函数——将符号多项式转化为数值多项式

【语法说明】 c = sym2poly(s)：返回符号多项式 s 的数值系数行向量 c。多项式自变量次数的系数按降幂排列。即行向量 c 的第一分量 c1 为多项式 s 的最高次数项的系数，c2 为第二高次数项的系数，依次类推。

【功能介绍】 将符号多项式转化为数值多项式。

【实例 4.58】 将符号多项式转换为数值多项式。

```
>>syms x u;
>>c1 = sym2poly(3*x^3 - 2*x^2 - sqrt(5))
>>c2 = sym2poly(u^4 - 3 + 5*u^2)
```

计算结果为：

```
c1 =
     3.0000   -2.0000     0   -2.2361
c2 =
     1     0     5     0    -3
```

【实例讲解】 这个例子中得到的结果是符号多项式按将幂排列的系数。

4.5.13 ccode 函数——符号表达式的 C 语言代码

【语法说明】 ccode(s)：返回 C 语言的、用于计算符号表达式 s 的语句。

【功能介绍】 返回符号表达式的 C 语言代码。

【实例 4.59】 将泰勒级数变换为 C 语言代码。

```
>>syms x
>>s = taylor(exp(x));
>>ccode(s)
```

计算结果为：

```
ans =
      t0    =   1.0+x+x*x/2.0+x*x*x/6.0+x*x*x*x/24.0+x*
x*x*x*x/120.0;
```

【实例讲解】 上面得到的结果为 e^x 在 x=0 附近的计算公式，即泰勒级数展开式。

4.5.14 fortran 函数——符号表达式的 Fortran 语言代码

【语法说明】 fortan(s)：返回 Fortan 语言的、用于计算符号表达式 s 的语句。

【功能介绍】 返回符号表达式的 Fortan 语言代码。

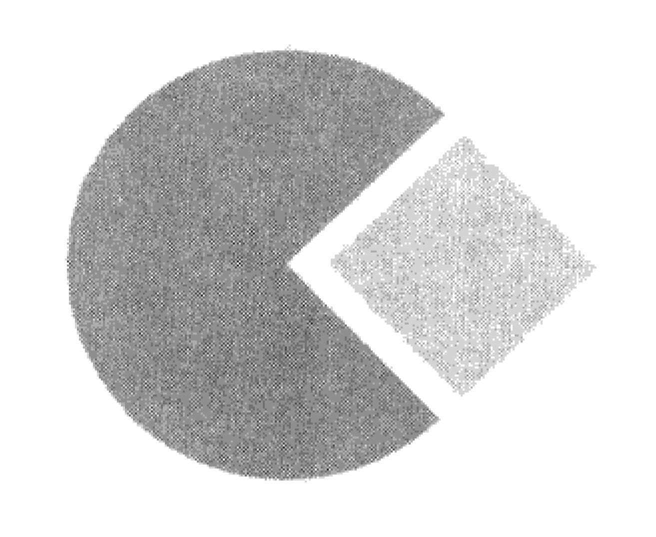
【实例 4.60】 显示 MATLAB 中对应的 Fortan 代码信息。

```
>>syms x
>>f = taylor(sin(x));
>>F1 = fortran(f)
>>H = sym(hilb(4));
>>F2 = fortran(t*(H))
```

计算结果为：

```
F1 =
        t0 = x-x**3/6+x**5/120
F2 =
        T(1,1)=t T(1,2)=t/2 T(1,3)=t/3 T(1,4)=t/4
        T(2,1)=t/2 T(2,2)=t/3 T(2,3)=t/4 T(2,4)=t/5
        T(3,1)=t/3 T(3,2)=t/4 T(3,3)=t/5 T(3,4)=t/6
        T(4,1)=t/4 T(4,2)=t/5 T(4,3)=t/6 T(4,4)=t/7
```

【实例讲解】 在 MATLAB 中，提供 fortan 函数的目的在于和 Fortan 语言的相互交流。



第 5 章　概率统计

概率统计是数学中极为关键的内容之一。利用 MATLAB 的强大工具来解决概率统计的数学问题，无论对于理论研究还是对于工程问题都是非常重要的。本章将围绕概率统计相关函数展开讨论。

5.1　随机数的产生

在研究概率统计之前，首先要生成一定的随机数，包括各种不同分布形式的随机数。产生随机数是概率统计得以继续进行的基础，下面的小节将就这方面的函数加以介绍。

5.1.1　binornd 函数——二项分布的随机数据的产生

【语法说明】

- R = binornd(N,P)：N、P 为二项分布的参数，返回服从参数为 N、P 的二项分布的随机数。
- R = binornd(N,P,m)：m 用来指定随机数的个数，它与 R 同维。
- R = binornd(N,P,m,n)：m、n 分别表示 R 的行数和列数。

【功能介绍】 参数为 N、P 的二项随机数据的产生。

【实例 5.1】 针对上面的语法说明产生几个随机数据。

```
>> R1=binornd(10,0.5)
R1 =
     3
>>   R2=binornd(10,0.5,3,6)

R2 =

     4     2     4     7     6     1
     5     4     4     6     6     3
     5     5     4     7     3     6

>>   R3=binornd(9,0.6,[5,10])

R3 =

     6     6     5     5     6     4     5     7     6     6
     3     4     3     7     5     6     6     7     5     8
     8     4     5     4     5     7     6     6     6     6
     5     4     4     5     6     6     5     4     3     8
     3     4     5     4     6     5     6     7     4     7
>>n = 10:10:60;
>>r1 = binornd(n,1./n)
r1 =
     2     1     0     1     1     2
>>r2 = binornd(n,1./n,[1 6])
r2 =
     0     1     2     1     3     1
```

【实例讲解】 R2 和 R3 是两种不同的指定维数的随机数据。

5.1.2 normrnd 函数——正态分布的随机数据的产生

【语法说明】

- R = normrnd(ave,sig)：返回均值为 ave，标准差为 sig 的正态分布的随机数据，R 可以是向量或矩阵。
- R = normrnd(ave,sig,m)：m 指定随机数的个数，与 R 同维数。
- R = normrnd(ave,sig,m,n)：m、n 分别表示 R 的行数和列数。

【功能介绍】 参数为μ、σ的正态分布的随机数据。

【实例 5.2】 利用 normrnd 函数产生几个正态分布的随机向量。

```
>> X1 = normrnd(1:6,1./(1:6))

X1 =

   0.5674   1.1672   3.0418   4.0719   4.7707   6.1985

>> X2 = normrnd(0,1,[1 6])

X2 =

   1.1892  -0.0376   0.3273   0.1746  -0.1867   0.7258

>> X3 = normrnd([2 3 4;8 9 10],0.1,2,3)   %ave 为均值矩阵

X3 =

   1.9412    2.9864    4.1067
   8.2183    9.0114   10.0059

>> X4 =normrnd(6,0.6,[3,3])   %ave 为 10,sig 为 0.5 的 2 行
3 列的正态随机数

X4 =

   5.5849    5.0438    5.7601
   6.5148    5.1354    6.4140
   6.7524    6.3427    6.4894
```

【实例讲解】 正态分布的数据可以是单个数值，如 X1、X2；也可以是矩阵形式的，如 X3、X4。

5.1.3 random 函数——通用函数求各分布的随机数据

【语法说明】 y = random('name',A1,A2,A3,m,n)：A1、A2、A3 为分布的参数，m、n 用来指定随机数的行和列，name 的取值有相关的表格来参照。

【功能介绍】 求各分布的随机数据。

【实例 5.3】 产生一个 5 行 6 列均值为 2，标准差为 0.5 的正态分布随机数。

```
>> y=random('norm',2,0.3,3,4)
```

计算结果为：

```
y =
    2.3567    2.0524    1.8235    2.0342
    1.9887    1.9440    2.6550    2.3200
2.0982    2.2177    1.9591    2.0178
```

【实例讲解】 random('norm',2,0.3,3,4)产生一个 3 行 4 列，均值为 2，标准差 0.3 的随机矩阵，从得到的结果可见，这些随机数都集中在 2 附近。

5.2　随机变量的描述

作为一种数据的表达形式，随机变量有其自己的描述方式，或者称为随机变量的特征。可以通过概率密度、概率值等来描述。

5.2.1　pdf 函数——通用函数计算概率密度函数值

【语法说明】

- Y=pdf(name,K,A)。
- Y=pdf(name,K,A,B)。
- Y=pdf(name,K,A,B,C)。

上面 3 个语法返回在 X=K 处，参数为 A、B、C 的概率密度值，对于不同的分布，参数个数不同；name 为分布函数名，其取值如下所示。

- 'beta'：β 分布。
- 'bino'：二项分布。
- 'chi2'：卡方分布。
- 'exp'：指数分布。
- 'f'：F 分布。
- 'gam'：γ 分布。
- 'geo'：几何分布。
- 'hyge'：超几何分布。

- 'logn'：对数正态分布。
- 'nbin'：负二项式分布。
- 'ncf'：非中心 *F* 分布。
- 'nct'：非中心 *T* 分布。
- 'ncx2'：非中心卡方分布。
- 'norm'：正态分布。
- 'poiss'：泊松分布。
- 't'：*T* 分布。
- 'unid'：离散均匀分布。
- 'weib'：Weibull 分布。

【功能介绍】 通用函数计算概率密度函数值。

【实例 5.4】 计算正态分布 *N*（0，1）的随机变量 *X* 在点 0.1245 处的密度函数值。

```
>> pdf('norm',.1245,0,1)
```

计算结果为：

```
ans =

    0.3959
```

【实例讲解】 用户可从正态分布的相关书籍，查看对应的结果。

5.2.2 binopdf 函数——二项分布的密度函数

【语法说明】 binopdf(x,n,p)：参数为 n、p 的二项分布的概率密度函数值。

【功能介绍】 求二项分布的密度函数。

【实例 5.5】 绘制以（9，0.3）为参数的二项分布的密度函数，并绘制其图形。

```
>> x = 0:20;
>> y = binopdf(x,9,0.3);
>> plot(x,y,':')
```

上例得到的结果如图 5.1 所示。

【实例讲解】 用户可以修改绘图的参数，查看相应的结果。

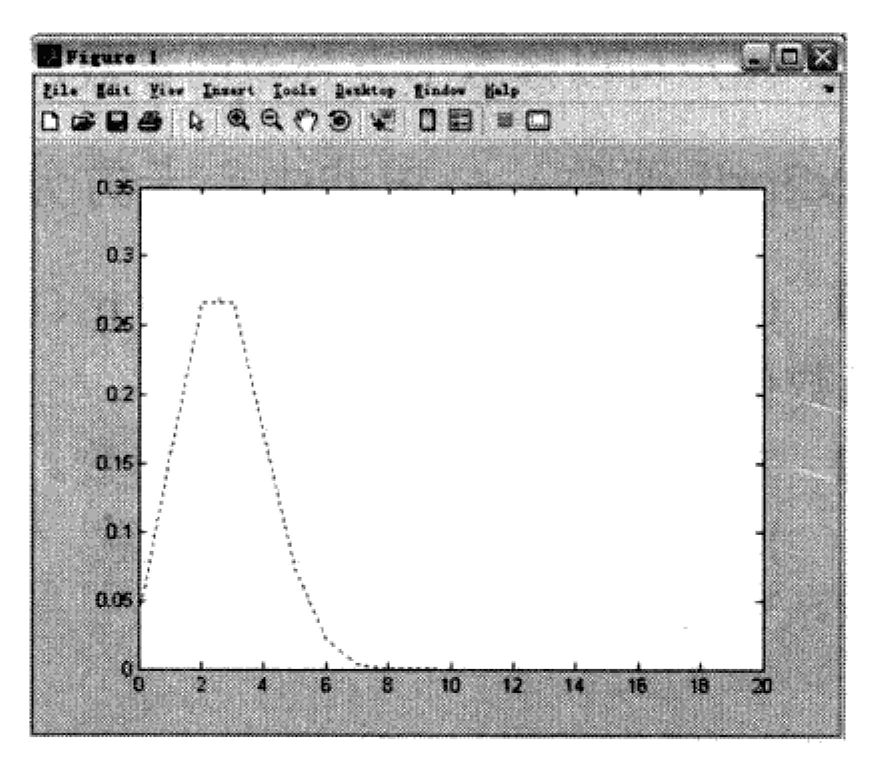


图 5.1　二项分布的密度函数

5.2.3　chi2pdf 函数——求卡方分布的概率密度函数

【语法说明】 chi2pdf(x, n)：自由度为 n 的卡方分布概率密度函数值。

【功能介绍】 求卡方分布的概率密度函数。

【实例 5.6】 绘制卡方分布的概率密度函数。

```
>> x = 0:0.2:15;
>>y = chi2pdf(x,4);
>>plot(x,y)
```

上例得到的图形如图 5.2 所示。

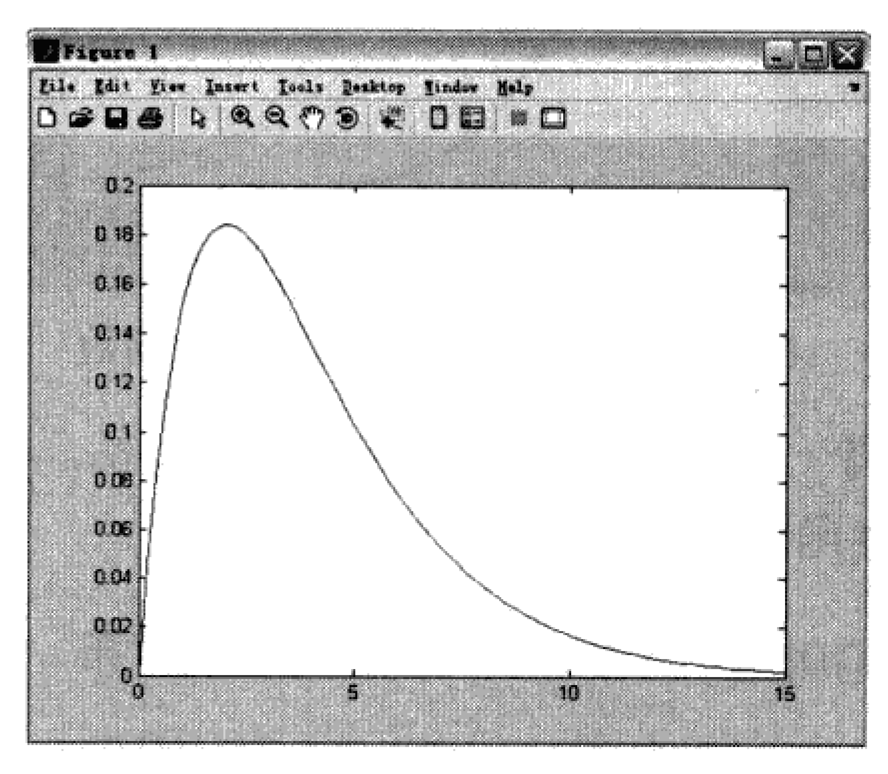


图 5.2　卡方分布的密度函数

【实例讲解】 用户可以修改卡方分布的参数，查看对比效果。

5.2.4 ncx2pdf 函数——求非中心卡方分布的密度函数

【语法说明】 ncx2pdf(x, n, delta)：参数为 n 、delta 的非中心 *F* 分布概率密度函数值。

【功能介绍】 求非中心卡方分布的密度函数。

【实例 5.7】 求非中心卡方分布的密度函数，绘制图形。

```
>> x = (0:0.2:20)';
>> y1 = ncx2pdf(x,6,3);
>> ynew= chi2pdf(x,4);
>> plot(x,y1,'-',x,ynew,'+')
```

上例得到的图形如图 5.3 所示。

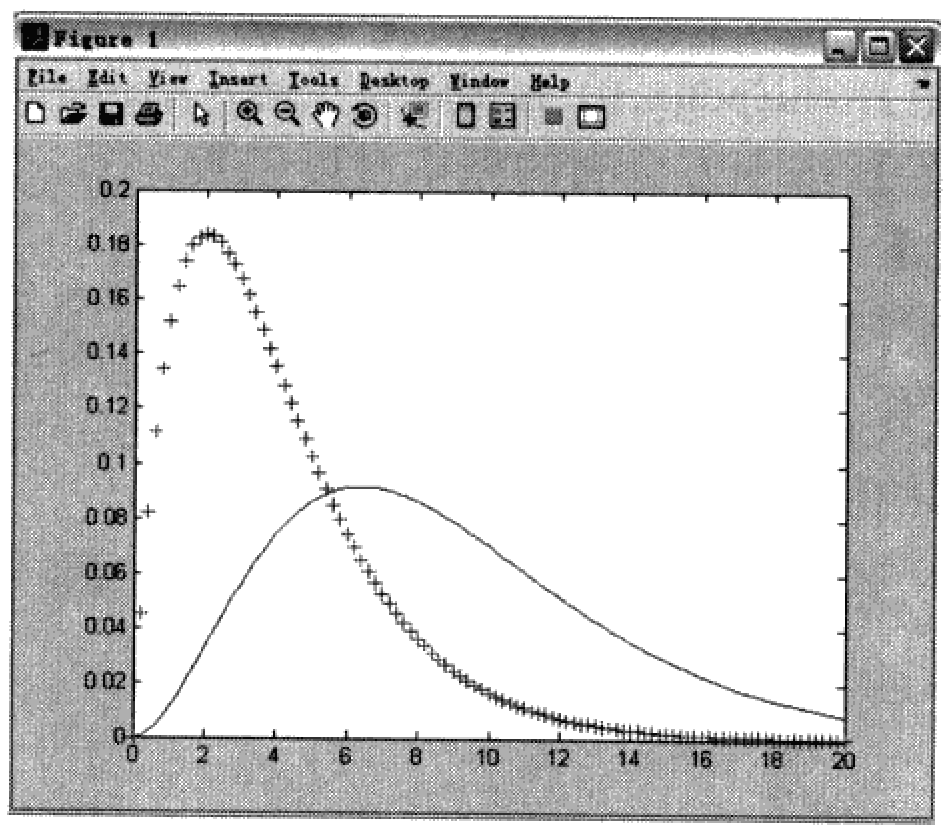


图 5.3 非中心卡方分布的密度函数

【实例讲解】 x 为横坐标的区间范围和步长，y 为 x 的非中心卡方分布的密度函数。

5.2.5 lognpdf 函数——对数正态分布

【语法说明】 lognpdf(x, mu, sigma)：参数为 mu、sigma 的对数

正态分布概率密度函数值。

【功能介绍】 求对数正态分布。

【实例 5.8】 产生对数正态分布数值，并绘制分布图。

```
>>x = (10:100:12800)';
>>y = lognpdf(x,log(10000),1.0);
>>plot(x,y)
```

上例得到的图形如图 5.4 所示。

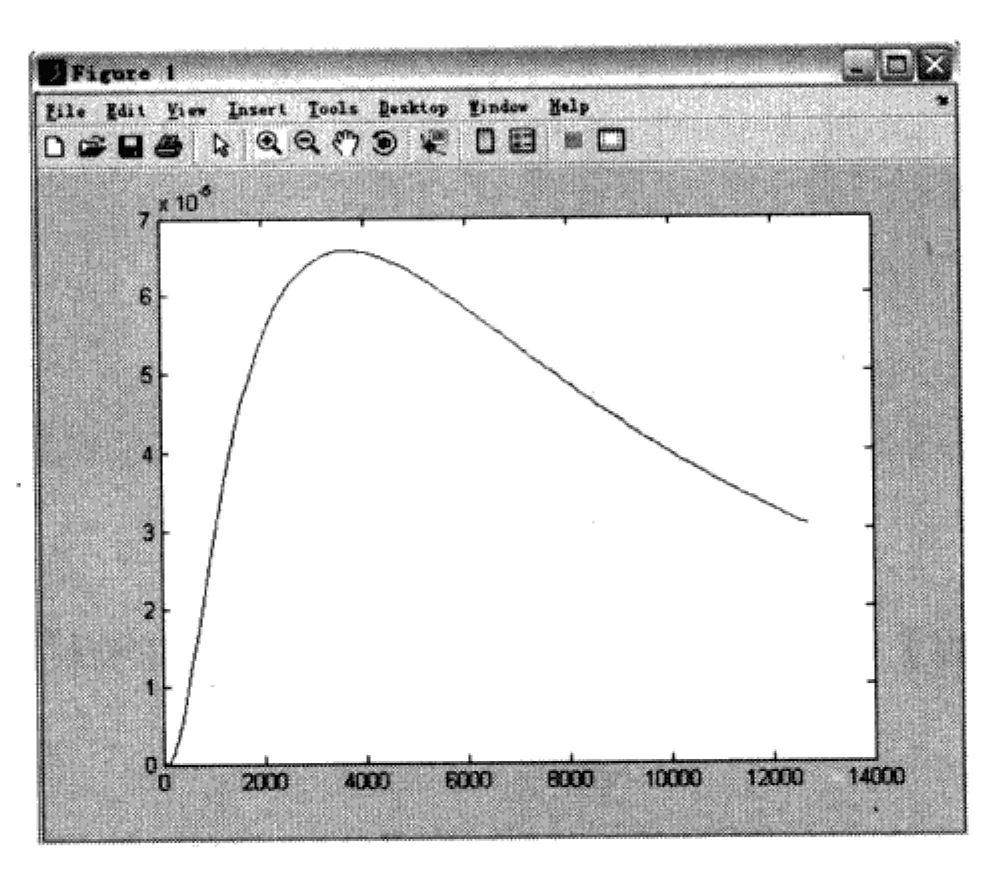


图 5.4　对数分布函数

【实例讲解】 x 为横坐标的区间范围和步长，y 为 x 的对数正态分布。

5.2.6　fpdf 函数——*F* 分布

【语法说明】 fpdf(x, n_1, n_2)：第一自由度为 n_1，第二自由度为 n_2 的 *F* 分布概率密度函数值。

【功能介绍】 求 *F* 分布概率密度函数。

【实例 5.9】 产生 *F* 分布数据，并绘制分布图。

```
>>x = 0:0.02:20;
>>y = fpdf(x,6,3);
>>plot(x,y)
```

上例得到的图形如图 5.5 所示。

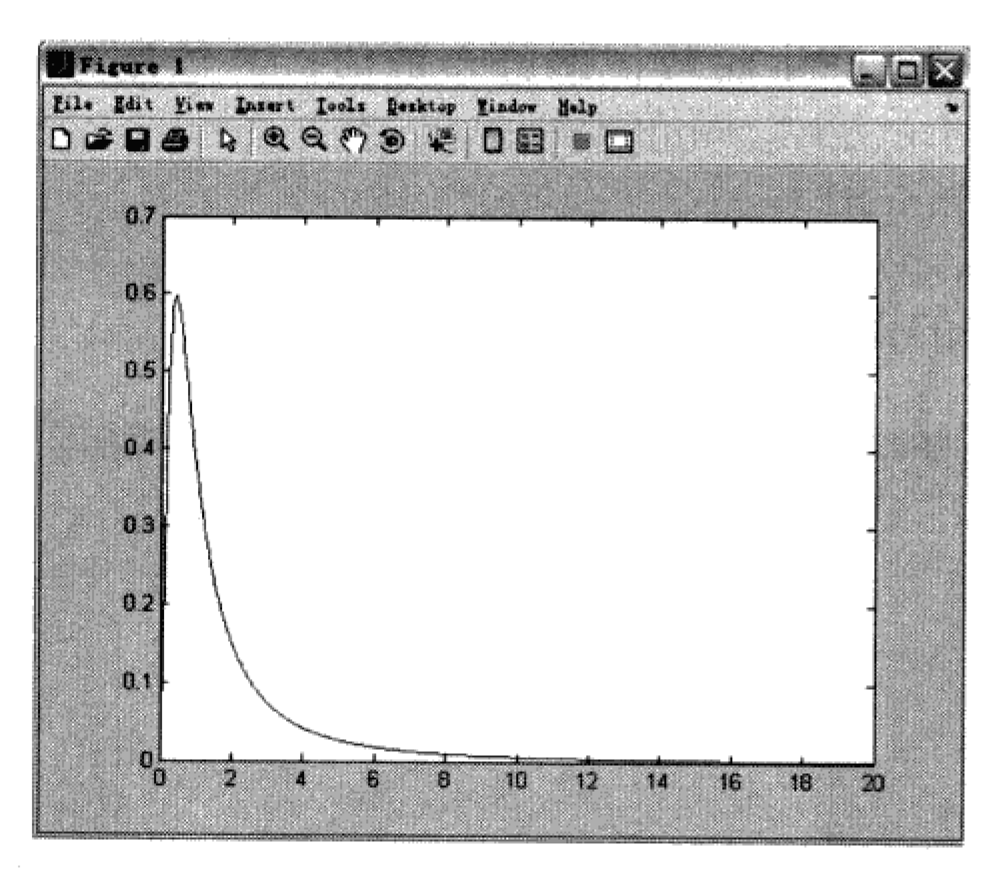


图 5.5　*F* 分布函数图形

【实例讲解】 x 定义了横坐标的区间范围和步长，y 为 x 的 *F* 分布。

5.2.7　ncfpdf 函数——求非中心 *F* 分布函数

【语法说明】 ncfpdf(x, n_1, n_2,)：参数为 n_1、n_2，delta 的非中心 *F* 分布概率密度函数值 fpdf(x, n_1, n_2)。

【功能介绍】 求非中心 *F* 分布函数。

【实例 5.10】 产生 *F* 分布数据，并给制非中心 *F* 分布图。

```
>>x = (0.01:0.1:10.01)';
>>p1 = ncfpdf(x,4,24,12);
>>p  = fpdf(x,5,20);
>>plot(x,p,'-',x,p1,'+')
```

上例得到的图形如图 5.6 所示。

【实例讲解】 x 定义了横坐标的区间范围和步长，y 为 x 的非中心 *F* 分布。

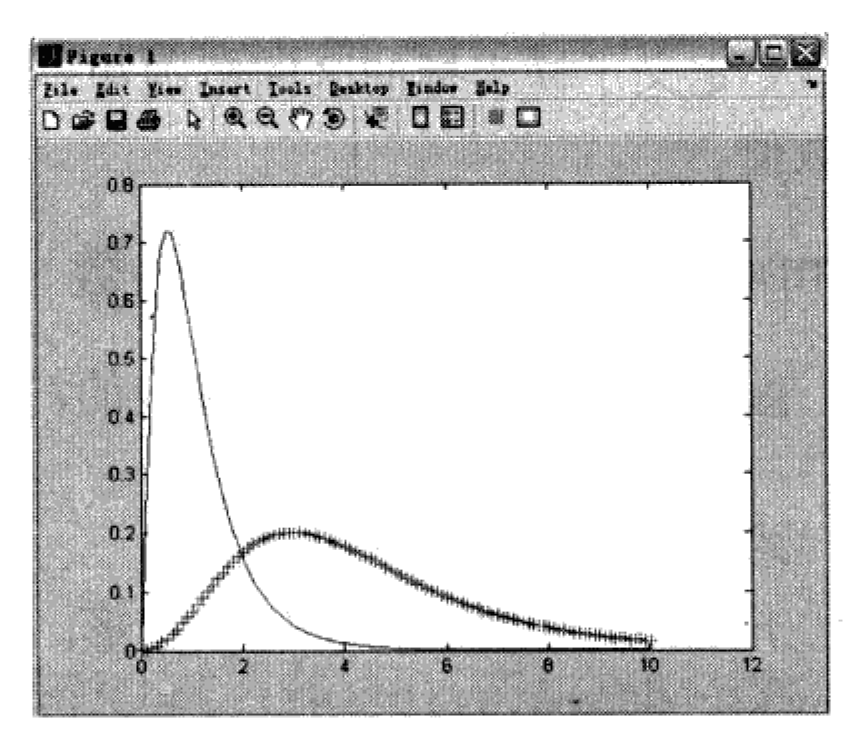


图 5.6 非中心 *F* 分布函数图

5.2.8 tpdf 函数——求 *T* 分布

【语法说明】 tpdf(x, n)：自由度为 n 的 *T* 分布概率密度函数值。

【功能介绍】 求 *T* 分布函数。

【实例 5.11】 产生 *T* 分布数据，并绘制 *T* 分布图。

```
>>x = -6:0.1:6;
>>y = tpdf(x,6);
>>z = normpdf(x,0,1);
>>plot(x,y,'-',x,z,'+')
```

上例得到的图形如图 5.7 所示。

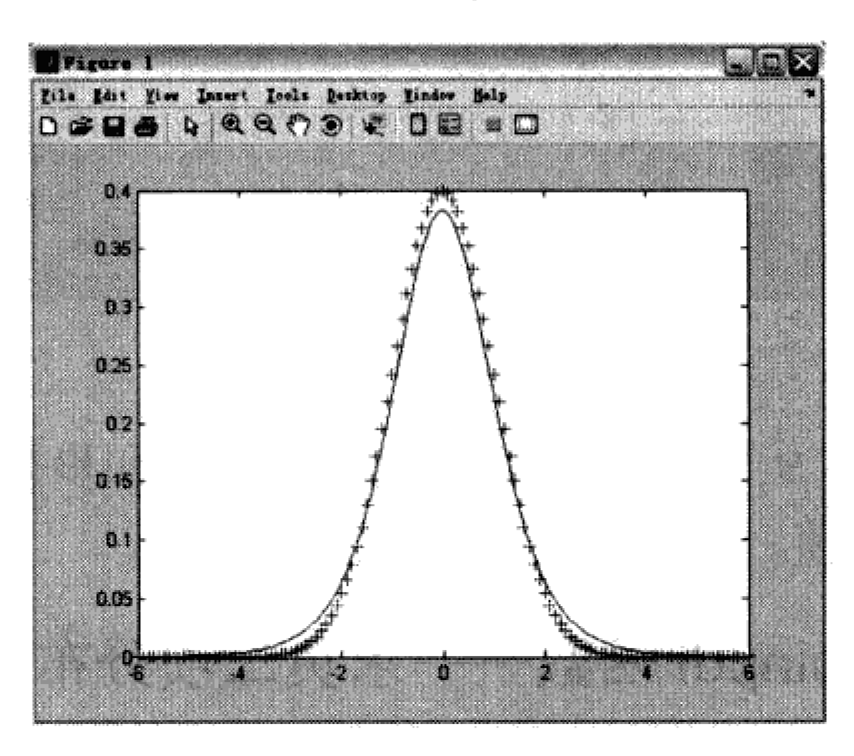


图 5.7 *T* 分布函数图

【实例讲解】 x 定义了横坐标的区间范围和步长，y 为 x 的 *T* 分布概率密度函数值。

5.2.9 gampdf 函数——求 Γ 分布函数

【语法说明】 gampdf(x, a, b)：参数为 a, b 的Γ分布概率密度函数值。

【功能介绍】 求Γ分布函数。

【实例 5.12】 产生Γ分布数据，并绘制图形。

```
>>x = gaminv((0.005:0.01:0.995),100,10);
>>y = gampdf(x,100,10);
>>ynew = normpdf(x,1000,100);
>>plot(x,y,'-',x,y1,'+')
```

上例得到的图形如图 5.8 所示。

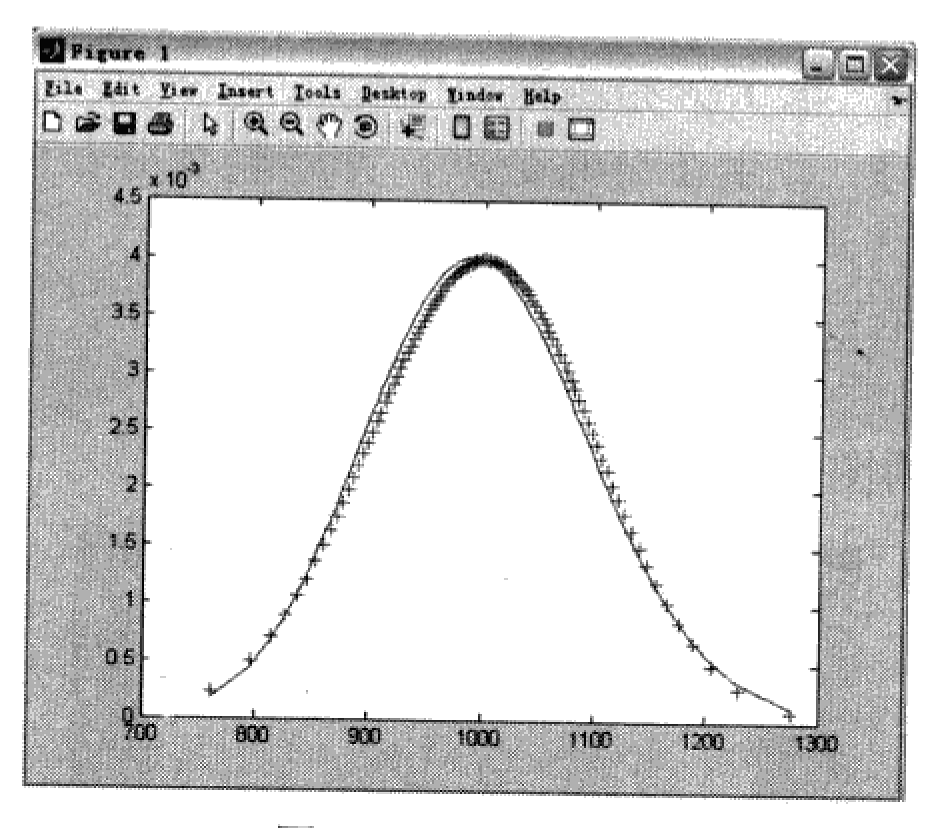


图 5.8 Γ分布函数

【实例讲解】 x 定义了横坐标的区间范围和步长，y 为 x 的Γ分布概率密度函数值。

5.2.10 nbinpdf 函数——求负二项分布

【语法说明】 nbinpdf(x, R, P)：参数为 R、P 的负二项式分布概

率密度函数值。

【功能介绍】 求负二项分布。

【实例 5.13】 产生负二项分布数据，并绘制图形。

```
>>x = (0:20);
>>y = nbinpdf(x,6,0.5);
>>plot(x,y,'--')
```

上例得到的图形如图 5.9 所示。

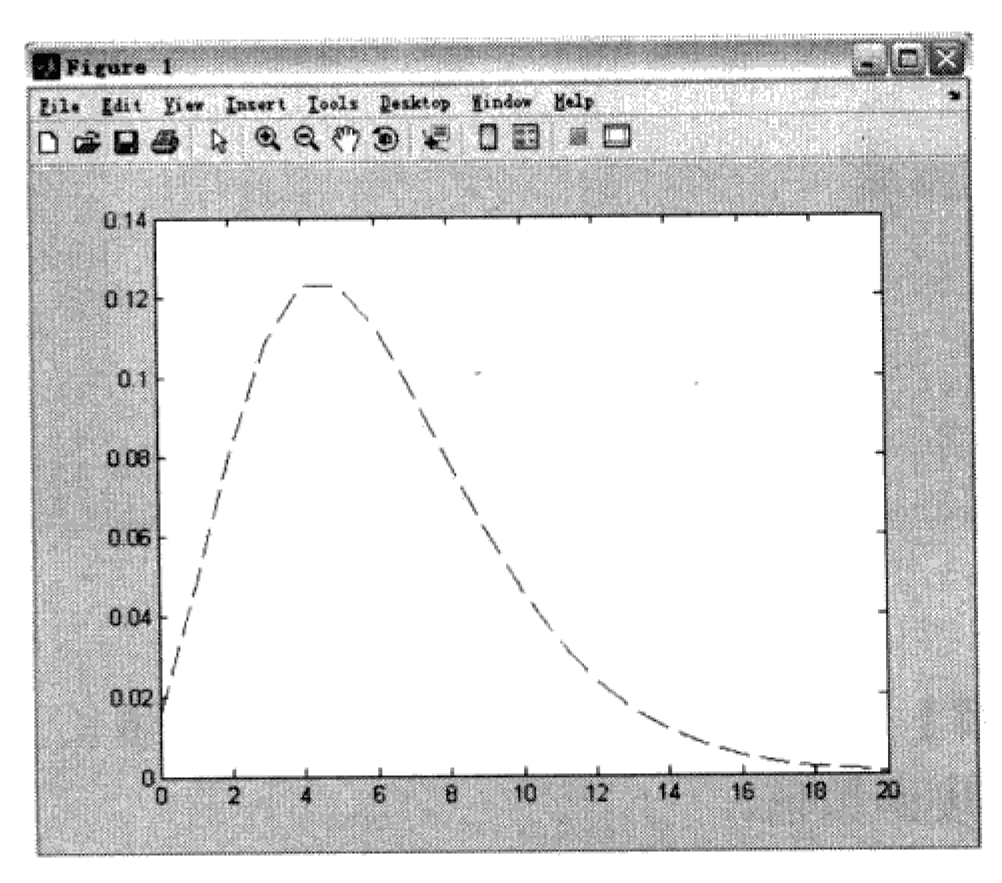


图 5.9 负二项分布函数

【实例讲解】 x 定义了横坐标的区间范围和步长，y 为 x 的负二项式分布概率密度函数值。

5.2.11 exppdf 函数——指数分布函数

【语法说明】 exppdf(x, Lambda)：参数为 Lambda 的指数分布概率密度函数值。

【功能介绍】 求指数分布函数。

【实例 5.14】 产生指数分布函数数据，并绘制图形。

```
>> x = 0:0.2:20;
>> y = exppdf(x,3);
>> plot(x,y)
```

上例得到的图形如图 5.10 所示。

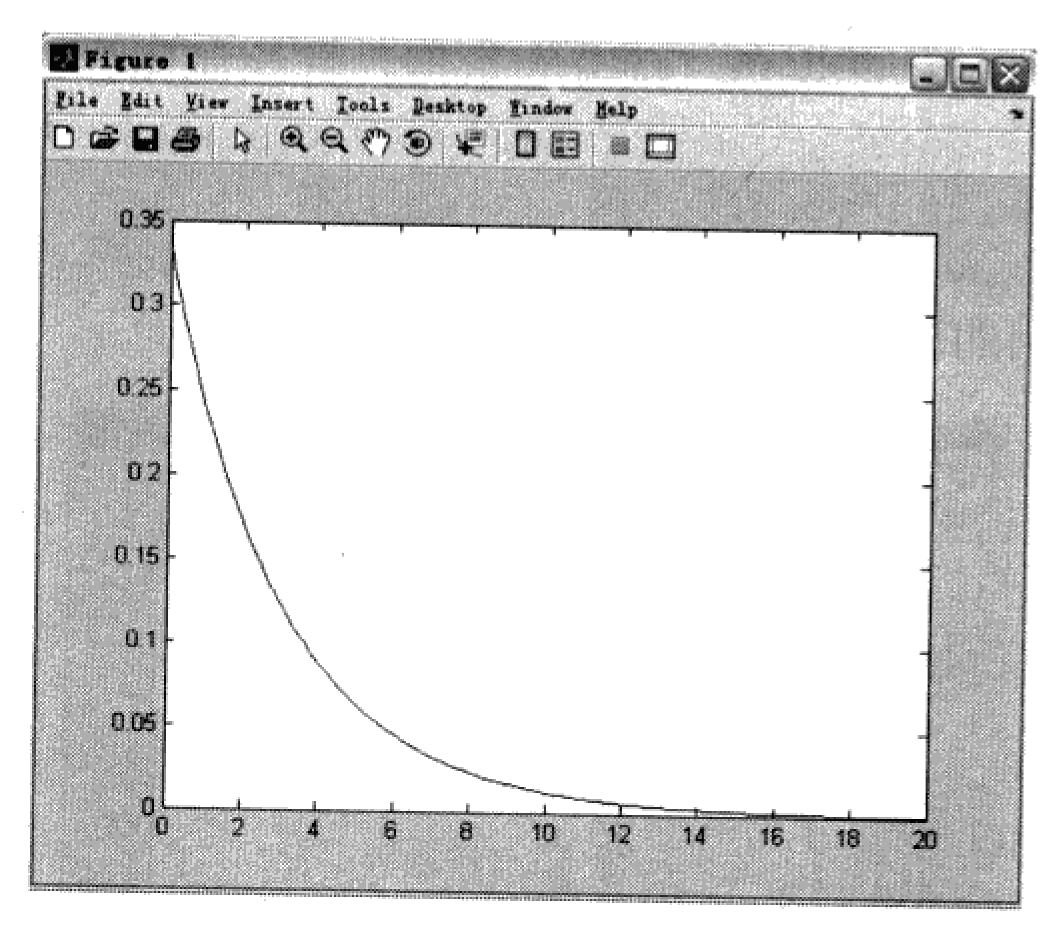


图 5.10　指数分布函数图形

【实例讲解】 x 定义了横坐标的区间范围和步长，y 为 x 的指数分布概率密度函数值。

5.2.12　raylpdf 函数——瑞利分布

【语法说明】 raylpdf(x, b)：参数为 b 的瑞利分布概率密度函数值。

【功能介绍】 求瑞利分布。

【实例 5.15】 产生瑞利分布的数据，并绘制图形。

```
>>x = [0:0.01:6];
>>p = raylpdf(x,0.6);
>>plot(x,p)
```

上例得到的图形如图 5.11 所示。

【实例讲解】 x 定义了横坐标的区间范围和步长，求得的 y 是基于 x 的参数为 6 的瑞利分布概率密度函数值。

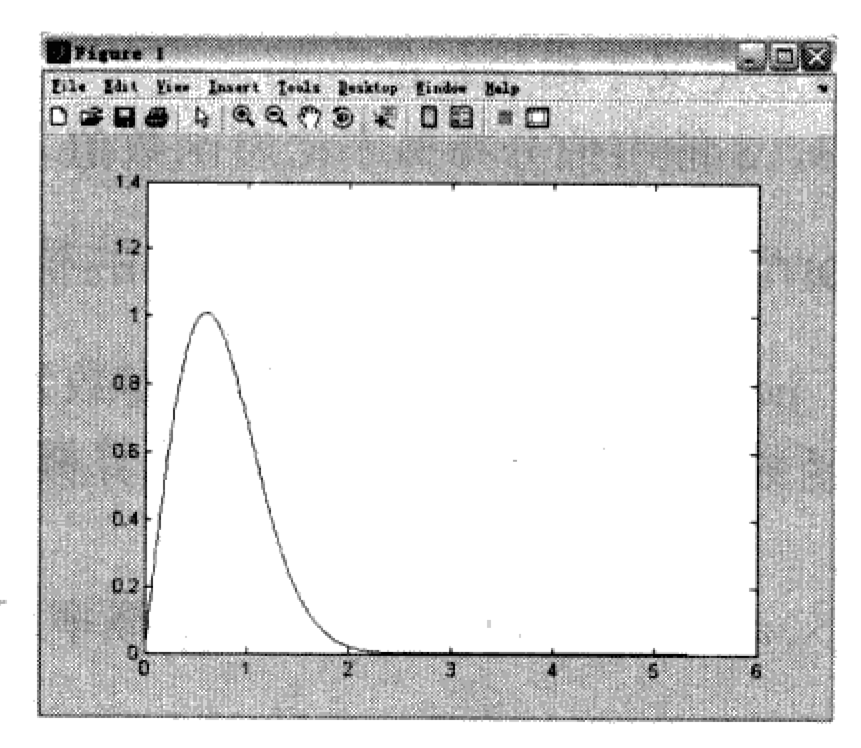


图 5.11　瑞利分布函数图

5.2.13　weibpdf 函数——求韦伯分布

【语法说明】 weibpdf(x, a, b)：参数为 a、b 的韦伯分布概率密度函数值。

【功能介绍】 求韦伯分布函数。

【实例 5.16】 产生韦伯分布的数据，并绘制图形。

```
>> x=0:0.1:6;
>> y=weibpdf(x,3,3);
>> plot(y)
```

上例得到的图形如图 5.12 所示。

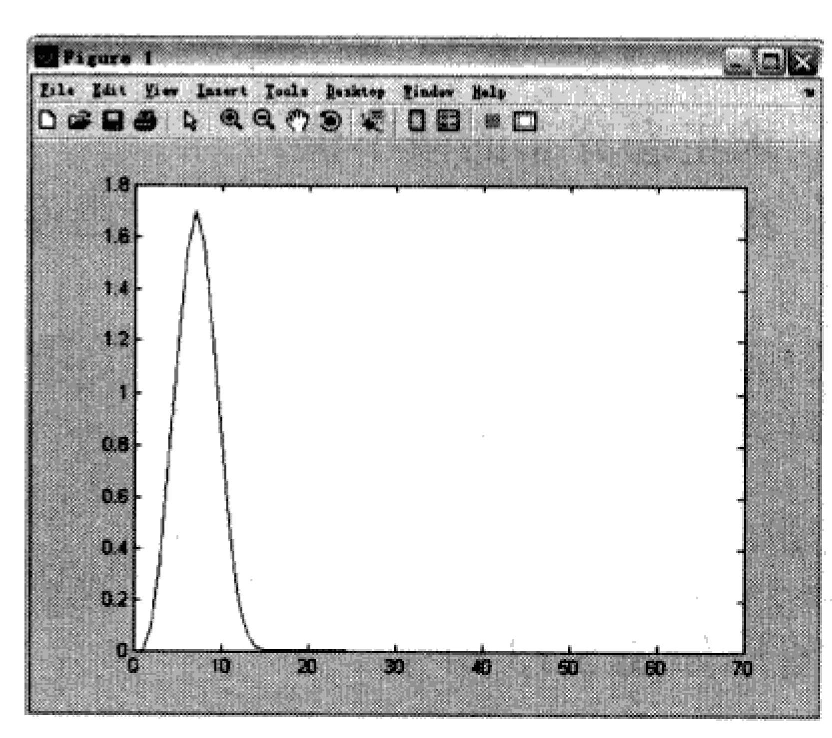


图 5.12　韦伯分布函数

【实例讲解】 x 定义了横坐标的区间范围和步长，求得的 y 是基于 x 的参数为 3，3 的韦伯分布概率密度函数值。

5.2.14 normpdf 函数——正态分布的概率值

【语法说明】 normpdf(K,mu,sigma)：计算参数为μ=mu，σ=sigma 的正态分布密度函数在 K 处的值。专用函数计算概率密度函数调用形式如下所示。

- unifpdf (x, a, b)：在[a,b]上均匀分布的概率密度在 X=x 处的函数值。
- Unidpdf(x,n)：均匀分布（离散）概率密度函数值。
- normpdf(x, mu, sigma)：参数为 mu，sigma 的正态分布概率密度函数值。
- betapdf(x, a, b)：参数为 a, b 的β分布概率密度函数值。
- nctpdf(x, n, delta)：参数为 n，delta 的非中心 T 分布概率密度函数值。
- geopdf(x,p)：参数为 p 的几何分布的概率密度函数值。
- hygepdf(x,M,K,N)：参数为 M，K，N 的超几何分布的概率密度函数值。
- poisspdf(x,Lambda)：参数为 Lambda 的泊松分布的概率密度函数值。

【功能介绍】 求正态分布的概率值。

【实例 5.17】 绘制卡方分布密度函数在自由度分别为 1、3、10 的图形。

```
>> x=0:0.1:20;
>>  y1=chi2pdf(x,1); plot(x,y1,'+')
>> hold on
>>  y2=chi2pdf(x,3);plot(x,y2,'.')
>> y3=chi2pdf(x,10);plot(x,y3,':')
```

上例得到的图形如图 5.13 所示。

【实例讲解】

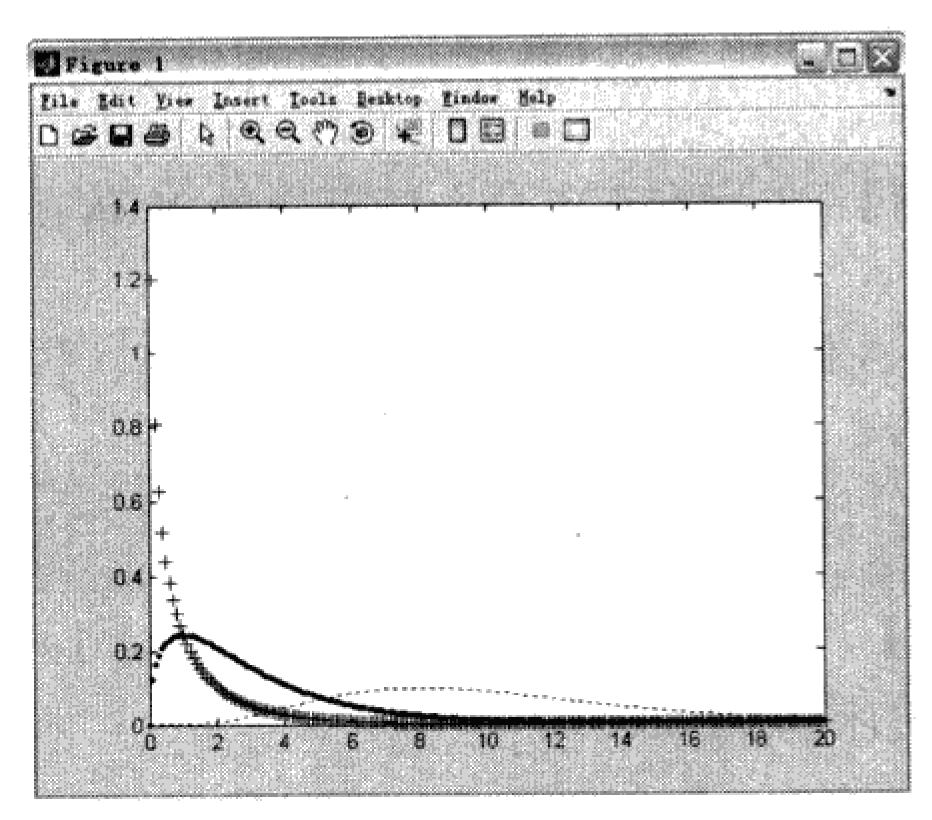


图 5.13　不同自由度的卡方分布密度函数图

5.2.15　poisspdf 函数——泊松分布的概率值

【语法说明】 poisspdf(k, Lamda)：等同于 pdf('poiss',K,Lamda)。

【功能介绍】 泊松分布的概率值。

5.3　随机变量的累积概率

在一些工业生产计算中，只知道随机变量的累积概率密度是不够的，通常需要计算某个区间内的统计概率值，这就是累计概率的计算问题。下面的章节介绍如何求取累计概率值。

5.3.1　cdf 函数——通用函数计算累积概率

【语法说明】

- cdf('name',K,A)：返回以 name 为分布，随机变量 X≤K 的概率之和的累积概率值。
- cdf('name',K,A,B)：A、B 为随机变量的两个参数。
- cdf('name',K,A,B,C)：A、B、C 为随机变量的 3 个参数。

【功能介绍】 通用函数 cdf 用来计算随机变量 X≤K 的概率之和（累积概率值）。

【实例 5.18】 求标准正态分布随机变量 X 落在区间(−∞，0.4)内的概率。

```
>> cdf('norm',0.4,0,1)
```

计算结果为：

```
ans =
    0.6554
```

【实例讲解】 这个例子中求取了服从 N（0，1）分布的随机变量在 $X\leqslant 0.4$ 内的概率是 0.6554。

5.3.2 binocdf 函数——二项分布的累积概率值

【语法说明】 binocdf (k, n, p)：n 为试验总次数，p 为每次试验事件 A 发生的概率，k 为 n 次试验中事件 A 发生的次数，该函数返回 n 次试验中事件 A 恰好发生 k 次的概率。

【功能介绍】 二项分布的累积概率值。

5.3.3 normcdf 函数——正态分布的累积概率值

【语法说明】 normcdf (x,mu,sigma)：返回 $F(x)=\int_{-\infty}^{x} p(t)\mathrm{d}t$ 的值，mu、sigma 为正态分布的两个参数。

【功能介绍】 求正态分布的累积概率值。

【实例 5.19】 设 X～N（3, 2^2）

（1）求 $P\{2<X<5\}$，$P\{-4<X<10\}$，$P\{|X|>2\}$，$P\{X>3\}$

（2）确定 c，使得 $P\{X>c\}=P\{X<c\}$

p1= $P\{2<X<5\}$

p2= $P\{-4<X<10\}$

p3= $P\{|X|>2\}=1-P\{|X|\leqslant 2\}$

p4= $P\{X>3\}=1-P\{X\leqslant 3\}$

```
>>p1=normcdf(5,3,2)-normcdf(2,3,2)
p1 =
    0.5328
>>p2=normcdf(10,3,2)-normcdf(-4,3,2)
p2 =
    0.9995
>>p3=1-normcdf(2,3,2)-normcdf(-2,3,2)
p3 =
    0.6853
>>p4=1-normcdf(3,3,2)
p4 =
    0.5000
```

【实例讲解】 专用函数计算累积概率值如下所示。

- unidcdf(x,n)：均匀分布（离散）累积分布函数值 $F(x)=P\{X\leqslant x\}$。
- unifcdf (x, a, b)：[a,b]上均匀分布（连续）累积分布函数值 $F(x)=P\{X\leqslant x\}$。
- ncfcdf(x, n_1, n_2, delta)：参数为 n_1、n_2、delta 的非中心 F 分布累积分布函数值。
- nctcdf(x, n, delta)：参数为 n_1、n_2、delta 的非中心 F 分布累积分布函数值。
- expcdf(x, Lambda)：参数为 Lambda 的指数分布累积分布函数值 $F(x)=P\{X\leqslant x\}$。
- nbincdf(x, R, P)：参数为 R、P 的负二项式分布概累积分布函数值 $F(x)=P\{X\leqslant x\}$。
- binocdf(x,n,p)：参数为 n、p 的二项分布的累积分布函数值 $F(x)=P\{X\leqslant x\}$。
- geocdf(x,p)：参数为 p 的几何分布的累积分布函数值 $F(x)=P\{X\leqslant x\}$。
- ncx2cdf(x, n, delta)：参数为 n、delta 的非中心卡方分布累积分布函数值。
- normcdf(x, mu, sigma)：参数为 mu、sigma 的正态分布累积

分布函数值 $F(x)=P\{X\leq x\}$。

- logncdf(x, mu, sigma)：参数为 mu、sigma 的对数正态分布累积分布函数值。
- chi2cdf(x, n)：自由度为 n 的卡方分布累积分布函数值 $F(x)=P\{X\leq x\}$。
- tcdf(x, n)：自由度为 n 的 T 分布累积分布函数值 $F(x)=P\{X\leq x\}$。
- fcdf(x, n_1, n_2)：第一自由度为 n_1，第二自由度为 n_2 的 F 分布累积分布函数值。
- gamcdf(x, a, b)：参数为 a、b 的γ分布累积分布函数值 $F(x)=P\{X\leq x\}$。
- betacdf(x, a, b)：参数为 a、b 的β分布累积分布函数值 $F(x)=P\{X\leq x\}$。
- raylcdf(x, b)：参数为 b 的瑞利分布累积分布函数值 $F(x)=P\{X\leq x\}$。
- weibcdf(x, a, b)：参数为 a、b 的韦伯分布累积分布函数值 $F(x)=P\{X\leq x\}$。
- hygecdf(x,M,K,N)：参数为 M、K、N 的超几何分布的累积分布函数值。
- poisscdf(x,Lambda)：参数为 Lambda 的泊松分布的累积分布函数值 $F(x)=P\{X\leq x\}$。

5.4 随机变量的逆累积分布函数

顾名思义，逆累计分布与累计分布是互逆的运算，逆累计分布主要是通过概率值来求取临界点。在一些工程运算中，这个临界点是非常有用的，所以在下面小节中讲解这样几个函数。

5.4.1　icdf 函数——计算逆累积分布函数

【语法说明】 icdf('name',P,a_1,a_2,a_3)：返回分布为 name，参数为 a_1、a_2、a_3，累积概率值为 P 的概率分布值。

【功能介绍】 计算逆累积分布函数。

【实例 5.20】 在χ^2分布表中，自由度为 9，$\alpha=0.975$，求临界值 Lamda。

因为表中给出的值满足$P\{\chi^2>\lambda\}=\alpha$，而逆累积分布函数 icdf 求满足$P\{\chi^2<\lambda\}=\alpha$的临界值$\lambda$。所以，$\alpha$取为 0.025，则有：

```
>> L=icdf('chi2',0.025,9)

L =

    2.7004
```

【实例讲解】 通过 icdf 可以直接求得临界值。

5.4.2　norminv 函数——正态分布逆累积分布函数

【语法说明】 X=norminv(p,mu,sigma)：p 为累积概率值，mu 为均值，sigma 为标准差，X 为临界值，满足 p=P{X≤x}。

【功能介绍】 正态分布逆累积分布函数。

【实例 5.21】 设$X\sim N(3,2^2)$，确定 c 使得$P\{X>c\}=P\{X<c\}$。

由$P\{X>c\}=P\{X<c\}$得，$P\{X>c\}=P\{X<c\}=0.5$，所以

```
>>X=norminv(0.5, 3, 2)
X=
   3
```

【实例讲解】 根据语法说明设置里面的参数值，可以得到临界值。

5.5　随机变量的数字特征

对随机变量的表示，除了概率密度、累积概率值等，还有随机

变量的数字特征对于其本身的表达也是非常重要的。这一节就要简单介绍与随机变量的数字特征有关系的一些知识。

5.5.1 sort 函数——排序

【语法说明】

- Y=sort(X)：X 为向量，返回 X 按由小到大排序后的向量。
- Y=sort(A)：A 为矩阵，返回 A 的各列按由小到大排序后的矩阵。
- [Y,I]=sort(A)：Y 为排序的结果，I 中元素表示 Y 中对应元素在 A 中位置。
- sort(X,dim)：在给定的维数 dim 内排序。若 X 为复数，则通过|X|排序。

【功能介绍】 对向量 X 按一定的规则排序。

【实例 5.22】 对矩阵 $A=\begin{bmatrix}1 & 2 & 3\\4 & 5 & 2\\3 & 7 & 0\end{bmatrix}$按一定的规则进行排序。

```
>> A=[1 2 3;4 5 2;3 7 0]
A =
     1     2     3
     4     5     2
     3     7     0
>> sort(A)
ans =
     1     2     0
     3     5     2
     4     7     3
>> [Y,I]=sort(A)
Y =
     1     2     0
     3     5     2
     4     7     3
I =
     1     1     3
```

```
     3     2     2
     2     3     1
```

【实例讲解】 这个例子中都是对实数值的排序，如果向量或矩阵中有复数，那么要通过|X|进行排序。

5.5.2　sortrows 函数——按行方式排序

【语法说明】

- Y=sortrows(A)：A 为矩阵，返回矩阵 Y，Y 按 A 的第 1 列由小到大，以行方式排序后生成矩阵。
- Y=sortrows(A, col)：按指定列 col 由小到大进行排序。
- [Y,I]=sortrows(A, col)：Y 为排序的结果，I 表示 Y 中第 col 列元素在 A 中位置。若 X 为复数，则通过|X|的大小排序。

【功能介绍】 按行方式进行排序。

【实例 5.23】 对矩阵 $A=\begin{bmatrix}1 & 2 & 3\\4 & 5 & 2\\3 & 7 & 0\end{bmatrix}$进行排序。

```
>> A=[1 2 3;4 5 2;3 7 0]
A =
     1     2     3
     4     5     2
     3     7     0
>> sortrows(A)
ans =
     1     2     3
     3     7     0
     4     5     2
>> sortrows(A,1)  %按第 1 列进行排序
ans =
     1     2     3
     3     7     0
     4     5     2
>> sortrows(A,3)     %按第 3 列进行排序
ans =
     3     7     0
     4     5     2
```

```
        1     2     3
>> sortrows(A,[3 2])
ans =
        3     7     0
        4     5     2
        1     2     3
>> [Y,I]=sortrows(A,3)    % Y 为排序结果,I 是 Y 中第 3 列元素
在 A 中位置
Y =
        3     7     0
        4     5     2
        1     2     3
I =
        3
        2
        1
```

【实例讲解】 sortrows(A)按第一列由小到大，sortrows(A,3)为按第 3 列由小到大，所以行的顺序就会调整，[Y,I]=sortrows(A,3)的讲解可以参见程序中的注释。

5.5.3 mean 函数——计算样本均值

【语法说明】

- Y=mean(X)：X 为向量，返回 X 的样本均值。
- Y=mean(A)：A 为矩阵，返回 A 的样本均值。

【功能介绍】 计算向量或矩阵的样本均值。

【实例 5.24】 在北京某大学的学生身高调查中，随机抽取 10 个男生，测得其身高如下：（身高：cm）174.5　165　180.6　174.5　179　163　175.3　190　174　177.9。试求这 10 名男生的平均身高。

```
>> X=[174.5  165  180.6  174.5  179  163  175.3  190
174 177.9]

X =

  Columns 1 through 8
```

```
   174.5000  165.0000  180.6000  174.5000  179.0000
163.0000  175.3000  190.0000

   Columns 9 through 10

   174.0000  177.9000

>> mean(X)   %计算样本均值

ans =

175.3800
```

【实例讲解】 这 10 名同学的平均身高是 175.38cm。

5.5.4 var 函数——求样本方差

【背景知识】 向量的样本方差定义为：$\mathrm{var}(X)= s^2 = \dfrac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{X})^2$ 。

【语法说明】

- D=var(X)：若 X 为向量，则返回向量的样本方差。
- D=var(A)：A 为矩阵，则 D 为 A 的列向量的样本方差构成的行向量。
- D=var(X, 1)：返回向量（矩阵）X 的简单方差（即置前因子为 $\dfrac{1}{n}$ 的方差）。
- D=var(X, w)：返回向量（矩阵）X 的以 w 为权重的方差。

【功能介绍】 求向量的样本方差。

【实例 5.25】 求下面 10 个男生同学身高的样本方差，（身高：cm）165　180.6　174.5　179　163　175.3　190　174　177.9 160。

```
>>  X=[165  180.6  174.5  179  163  175.3  190  174
177.9  160]

X =
```

```
    Columns 1 through 8

    165.0000   180.6000   174.5000   179.0000   163.0000
175.3000  190.0000  174.0000

    Columns 9 through 10

    177.9000  160.0000

  >> var(X)

  ans =

    82.1846
```

【实例讲解】 这10个男同学身高的样本方差是82.1846。

5.5.5 std函数——求标准差

【语法说明】

- std(X): 返回向量(矩阵)X的样本标准差(置前因子为$\frac{1}{n-1}$)

即 $std = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n} x_i - \bar{X}}$

- std(X,1): 返回向量（矩阵）X的标准差（置前因子为$\frac{1}{n}$）。
- std(X, 0): 与std (X)相同。
- std(X, flag, dim): 返回向量（矩阵）中维数为dim的标准差值，其中flag=0时，置前因子为$\frac{1}{n-1}$；否则置前因子为$\frac{1}{n}$。

【功能介绍】 求向量的标准差。

【实例5.26】 求下列样本的样本方差和样本标准差，方差和标准差。

165　180.6　174.5　179　163　175.3　190　174　177.9 160

```
>> X=[165 180.6 174.5 179 163 175.3 190 174
177.9 160]

X =

  Columns 1 through 8

   165.0000  180.6000  174.5000  179.0000  163.0000
175.3000  190.0000  174.0000

  Columns 9 through 10

  177.9000  160.0000

>> DX=var(X,1)        %求方差

DX =

   73.9661

>> sig=std(X,1)      %求标准差

sig =

    8.6004

>> DX1=var(X)        %求样本方差

DX1 =

   82.1846

>> sig1=std(X)      %求样本标准差

sig1 =

9.0656
```

【实例讲解】 从上面的结果中可以看出，样本的标准差和方差的差别。

5.5.6 nanstd 函数——忽略 NaN 计算的标准差

【语法说明】 y = nanstd(X)：若 X 为含有元素 NaN 的向量，则返回除 NaN 外的元素的标准差；若 X 为含元素 NaN 的矩阵，则返回各列除 NaN 外的标准差构成的向量。

【功能介绍】 忽略 NaN 计算标准差。

【实例 5.27】 首先产生一个魔方阵，将魔方阵中的某些元素置为 NaN，并在忽略 NaN 的情况下计算标准差。

```
>> M=magic(3)     %产生 3 阶魔方阵
M =
     8     1     6
     3     5     7
     4     9     2
>> M([1 6 8])=[NaN NaN NaN] %替换 3 阶魔方阵中第 1、6、8 个元素为 NaN
M =
   NaN     1     6
     3     5   NaN
     4   NaN     2
>> y=nanstd(M)                     %求忽略 NaN 的各列向量的标准差
y =
    0.7071    2.8284    2.8284
>> X=[1 5];                        %忽略 NaN 的第 2 列元素
>> y2=std(X)                       %验证第 2 列忽略 NaN 元素的标准差
y2 =
    2.8284
```

【实例讲解】 忽略 NaN 元素后，其原位置处的数值用 0 代替计算。

5.5.7 geomean 函数——计算几何平均数

【背景知识】 几何平均数不同于算术平均数，它的数学含义是 $M=(\prod_{i=1}^{n}x_i)^{\frac{1}{n}}$，在这里，样本数据为非负数值，常用于对数正态分布。

【语法说明】

- M=geomean(X)：X 为向量，返回 X 中各元素的几何平均数。
- M=geomean(A)：A 为矩阵，返回 A 中各列元素的几何平均数构成的向量。

【功能介绍】 计算几何平均数。

【实例 5.28】 计算几何平均数。

```
>> A=[1 2 3 4]

A =

     1     2     3     4

>> M=geomean(A)

M =

    2.2134

>>  B=[1 2 3 4;2 3 4 9;2 9 0 5]

B =

     1     2     3     4
     2     3     4     9
     2     9     0     5

>> M2=geomean(B)

M2 =

1.5874    3.7798         0    5.6462
```

【实例讲解】 对于一个向量，计算这个向量内几个数的几何平均数。对于矩阵而言，计算每一列的算术平均数，结果为新的向量。

5.5.8 mean 函数——求算术平均值

【语法说明】

- mean(X)：X 为向量，返回 X 中各元素的平均值。

- mean(A)：A 为矩阵，返回 A 中各列元素的平均值构成的向量。
- mean(A,dim)：返回给出的维数内的平均值。

【功能介绍】 利用 mean 求算术平均值。

【实例 5.29】 求矩阵 $A=\begin{bmatrix}1 & 3 & 4 & 5\\2 & 3 & 4 & 6\\1 & 3 & 1 & 5\\4 & 7 & 9 & 6\end{bmatrix}$ 的算术平均值。

```
>> A=[1 3 4 5;2 3 4 6;1 3 1 5;4 7 9 6]

A =

     1     3     4     5
     2     3     4     6
     1     3     1     5
     4     7     9     6

>> mean(A)

ans =

    2.0000    4.0000    4.5000    5.5000

>> mean(A,1)

ans =

    2.0000    4.0000    4.5000    5.5000

>> mean(A,3)

ans =

     1     3     4     5
     2     3     4     6
     1     3     1     5
     4     7     9     6
```

【实例讲解】 第一个函数是求矩阵中每一列的平均值。第二个

和第三个函数是给定的维数内的算术平均值。

5.5.9　nanmean 函数——忽略 NaN 元素计算算术平均值

【语法说明】

- nanmean(X)：X 为向量，返回 X 中除 NaN 外元素的算术平均值。
- nanmean(A)：A 为矩阵，返回 A 中各列除 NaN 外元素的算术平均值向量。

【功能介绍】 忽略 NaN 元素计算算术平均值。

【实例 5.30】 忽略矩阵中的 NaN 元素计算算术平均值。

```
>> A=[1 2 3;nan 5 2;3 7 nan]
A =
     1     2     3
   NaN     5     2
     3     7   NaN
>> nanmean(A)
ans =
    2.0000    4.6667    2.5000
```

【实例讲解】 除了 NaN 元素对其余的元素进行平均计算。

5.5.10　median 函数——计算中位数

【语法说明】

- median(X)：X 为向量，返回 X 中各元素的中位数。
- median(A)：A 为矩阵，返回 A 中各列元素的中位数构成的向量。
- median(A,dim)：求给出的维数内的中位数。

【功能介绍】 计算中位数。

【实例 5.31】 计算矩阵 $\boldsymbol{A}=\begin{bmatrix}1 & 2 & 4 & 6\\ 3 & 4 & 5 & 7\\ 8 & 9 & 6 & 0\\ 4 & 6 & 8 & 1\end{bmatrix}$ 中各列元素的中位数。

```
>> A=[1 2 4 6;3 4 5 7;8 9 6 0;4 6 8 1]

A =

     1     2     4     6
     3     4     5     7
     8     9     6     0
     4     6     8     1

>> median(A)

ans =

3.5000    5.0000    5.5000    3.5000
```

【实例讲解】 各个列元素的中位数重新构成新的矩阵。

5.5.11 nanmedian 函数——忽略 NaN 计算中位数

【语法说明】

- nanmedian(X)：X 为向量，返回 X 中除 NaN 外元素的中位数。
- nanmedian(A)：A 为矩阵，返回 A 中各列除 NaN 外元素的中位数向量。

【功能介绍】 忽略 NaN 计算中位数。

【实例 5.32】 忽略 NaN，计算中位数。

```
>> A=[1 2 3;nan 5 2;3 7 nan]
A =
     1     2     3
   NaN     5     2
     3     7   NaN
>> nanmedian(A)
ans =
2.0000    5.0000    2.5000
```

【实例讲解】 从上面的结果中可以看出，matlab 直接忽略 nan 数值。

5.5.12 harmmean 函数——求调和平均数

【背景知识】 调和平均值的数学含义是 $M=\dfrac{n}{\sum_{i=1}^{n}\dfrac{1}{x_i}}$，样本数据为非 0 数值，主要用于严重偏斜分布。

【语法说明】

- M=harmmean(X)：X 为向量，返回 X 中各元素的调和平均值。
- M=harmmean(A)：A 为矩阵，返回 A 中各列元素的调和平均值构成的向量。

【功能介绍】 计算调和平均值。

【实例 5.33】 求两个矩阵的调和平均值。

```
A =

     1     2     4     6
     3     4     5     7
     8     9     6     0
     4     6     8     1

>> M1=harmmean(A)
Warning: Divide by zero.
> In harmmean at 18

M1 =

    2.3415    3.8919    5.3933         0

>>  B=[1  6  0  9]

B =

     1     6     0     9

>> M2=harmmean(B)
Warning: Divide by zero.
```

```
> In harmmean at 18

M2 =

     0

>> C=[1  3  4  5]

C =

     1     3     4     5

>> M3=harmmean(C)

M3 =

    2.2430
```

【实例讲解】 如果求得的结果中有 0 元素，MATLAB 系统会给出提示信息。

5.5.13 range 函数——求最大值与最小值之差

【语法说明】

- Y=range(X)：X 为向量，返回 X 中的最大值与最小值之差。
- Y=range(A)：A 为矩阵，返回 A 中各列元素的最大值与最小值之差。

【功能介绍】 求最大值与最小值之差。

【实例 5.34】 求矩阵 $A=\begin{bmatrix}1 & 2 & 3\\2 & 8 & 9\\3 & 6 & 2\end{bmatrix}$ 中元素的最大值与最小值之差。

```
>> A=[1 2 3;2 8 9;3 6 2]

A =

     1     2     3
```

```
     2     8     9
     3     6     2

>> Y=range(A)

Y =

     2     6     7
```

【实例讲解】 通过结果可以看出，得到的结果是按照每一列进行运算的结果，如：第一列 3−1=2；第 2 列 8−2=6；第 3 列 9−2=7。

5.5.14 skewness 函数 —— 样本的偏斜度

【背景知识】 偏斜度是这样定义的：$y=\frac{E(x-\mu)^3}{\sigma^3}$，偏斜度样本数据是关于均值不对称的一个测度，如果偏斜度为负，说明均值左边的数据比均值右边的数据更散；如果偏斜度为正，说明均值右边的数据比均值左边的数据更散，因而正态分布的偏斜度为 0。

【语法说明】

- y = skewness(X)：X 为向量，返回 X 的元素的偏斜度；X 为矩阵，返回 X 各列元素的偏斜度构成的行向量。
- y = skewness(X,flag)：flag=0 表示偏斜纠正，flag=1（默认）表示偏斜不纠正。

其中：μ为 x 的均值，σ为 x 的标准差，E()为期望值算子。

【功能介绍】 求样本的偏斜度。

【实例 5.35】 计算样本的偏斜度。

```
>> X=randn([5,4])
X =
    0.2944    0.8580   -0.3999    0.6686
   -1.3362    1.2540    0.6900    1.1908
    0.7143   -1.5937    0.8156   -1.2025
    1.6236   -1.4410    0.7119   -0.0198
   -0.6918    0.5711    1.2902   -0.1567
>> y=skewness(X)
y =
```

```
   -0.0040   -0.3136   -0.8865   -0.2652
>> y=skewness(X,0)
y =
   -0.0059   -0.4674   -1.3216   -0.3954
```

【实例讲解】 通过 X=randn([5,4])产生 5 行 4 列的随机矩阵，利用 skewness（）函数对 X 矩阵的偏斜度进行计算。

5.5.15 unifstat 函数——均匀分布的期望和方差

【语法说明】 [M,V]=unifstat(A,B)：A、B 为标量时，就是区间上均匀分布的期望和方差，A、B 也可为向量或矩阵，则 M、V 也是向量或矩阵。

【功能介绍】 求均匀分布的期望和方差。

【实例 5.36】 计算均匀分布的期望和方差。

```
>>a = 1:6; b = 2.*a;
>>[M,V] = unifstat(a,b)
```

计算结果为：

```
M =
    1.5000    3.0000    4.5000    6.0000    7.5000    9.0000
V =
    0.0833    0.3333    0.7500    1.3333    2.0833    3.0000
```

【实例讲解】 得到的结果就是均匀分布的期望和方差。

5.5.16 normstat 函数——正态分布的期望和方差

【语法说明】 [M,V] = normstat(mu,sig)：mu、sig 可为标量也可为向量或矩阵。

【功能介绍】 求正态分布的期望和方差。

【实例 5.37】 计算正态分布的期望和方差。

```
>> n=1:4;
>> [M,V]=normstat(n'*n,n'*n)
```

计算结果为：

```
M =
     1     2     3     4
```

```
      2         4         6         8
      3         6         9         12
      4         8         12        16
V =
      1         4         9         16
      4         16        36        64
      9         36        81        144
      16        64        144       256
```

【实例讲解】 通过结果可以看出 V 中的每个元素均是 M 中元素的平方。

5.5.17 binostat 函数——二项分布的均值和方差

【语法说明】 [M,V] = binostat(N,P)：N、P 为二项分布的两个参数，可为标量也可为向量或矩阵。

【功能介绍】 求二项分布的均值和方差。

【实例 5.38】 计算二项分布的均值和方差。

```
>>n = logspace(1,5,5)
n =
          10        100        1000       10000       100000
>>[M,V] = binostat(n,1./n)
M =
     1     1     1     1     1
V =
    0.9000    0.9900    0.9990     0.9999    1.0000
>>[m,v] = binostat(n,1/2)
m =
           5         50        500        5000        50000
v =
   1.0e+04 *
    0.0003    0.0025    0.0250    0.2500    2.5000
```

【实例讲解】 常见分布的期望和方差的调用方式如下所示。

- [M,V]=unifstat (a, b) ：均匀分布（连续）的期望和方差，M 为期望，V 为方差。
- [M,V]=unidstat (n)：均匀分布（离散）的期望和方差。

- [M,V]=expstat (p, Lambda)：指数分布的期望和方差。
- [M,V]=normstat(mu,sigma)：正态分布的期望和方差。
- [M,V]=chi2stat (x, n)：卡方分布的期望和方差。
- [M,V]=tstat (n) ：*T* 分布的期望和方差。
- [M,V]=fstat (n_1, n_2) ：*F* 分布的期望和方差。
- [M,V]=gamstat (a, b)：γ分布的期望和方差。
- [M,V]=betastat (a, b)：β分布的期望和方差。
- [M,V]=lognstat (mu, sigma)：对数正态分布的期望和方差。
- [M,V]=nbinstat (R, P)：负二项式分布的期望和方差。
- [M,V]=ncfstat (n_1, n_2, delta)：非中心 *F* 分布的期望和方差。
- nctstat (n, delta)：非中心 *T* 分布的期望和方差。
- [M,V]=ncx2stat (n, delta)：非中心卡方分布的期望和方差。
- [M,V]=raylstat (b)：瑞利分布的期望和方差。
- [M,V]=weibstat (a, b)：韦伯分布的期望和方差。
- [M,V]=binostat (n,p)：二项分布的期望和方差。
- [M,V]=geostat (p) ：几何分布的期望和方差。
- [M,V]=hygestat (M,K,N)：超几何分布的期望和方差。
- [M,V]=poisstat (Lambda)：泊松分布的期望和方差。

5.5.18 cov 函数——协方差

【语法说明】

- cov(X)：求向量 X 的协方差。
- cov(A)：求矩阵 A 的协方差矩阵，该协方差矩阵的对角线元素是 A 的各列的方差，即 var(A)=diag(cov(A))。
- cov(X,Y)：X、Y 为等长列向量。

【功能介绍】 求向量的协方差。

【实例 5.39】 求向量的协方差。

```
>> X=[0 -1 1]';
>> Y=[1 2 2]';
```

```
>> C1=cov(X)          %X 的协方差
C1 =
     1
>> C2=cov(X,Y)  %列向量 X、Y 的协方差矩阵,对角线元素为各列向
量的方差
C2 =
    1.0000         0
         0    0.3333
>> A=[1 2 3;4 0 -1;1 7 3]
A =
     1     2     3
     4     0    -1
     1     7     3
>> C1=cov(A)          %求矩阵 A 的协方差矩阵
C1 =
    3.0000   -4.5000   -4.0000
   -4.5000   13.0000    6.0000
   -4.0000    6.0000    5.3333
>> C2=var(A(:,1))   %求 A 的第 1 列向量的方差
C2 =
     3
>> C3=var(A(:,2))   %求 A 的第 2 列向量的方差
C3 =
    13
>> C4=var(A(:,3))
C4 =
    5.3333
```

【实例讲解】 C2=cov(X,Y)用来求 X，Y 向量的协方差，C3=var(A(:,2))通过参数 2 指定列数，来求 A 矩阵第 2 列的协方差。

5.5.19　corrcoef 函数——相关系数

【语法说明】

- corrcoef(X,Y)：返回列向量 X、Y 的相关系数，等同于 corrcoef([X　Y])。
- corrcoef (A)：返回矩阵 A 的列向量的相关系数矩阵。

【功能介绍】 求相关系数。

【实例 5.40】 求矩阵 $A=\begin{bmatrix}1 & 3 & 5 & 7\\3 & 4 & 5 & 6\\6 & 6 & 9 & 0\\3 & 2 & 1 & 6\end{bmatrix}$ 的相关系数及列向量之间的相关系数。

```
>> A=[1 3 5 7;3 4 5 6;6 6 9 0;3 2 1 6]

A =

     1     3     5     7
     3     4     5     6
     6     6     9     0
     3     2     1     6

>> C1=corrcoef(A)

C1 =

    1.0000    0.7811    0.5941   -0.9469
    0.7811    1.0000    0.9562   -0.8687
    0.5941    0.9562    1.0000   -0.7651
   -0.9469   -0.8687   -0.7651    1.0000

>> C1=corrcoef(A(:,2),A(:,3))      %求A的第2列与第3列列向量的相关系数矩阵

C1 =

    1.0000    0.9562
    0.9562    1.0000
```

【实例讲解】 corrcoef(A)用来求矩阵本身的相关系数，C1=corrcoef(A(:,2),A(:,3))用来求矩阵 A 中第 2 列和第 3 列的相关系数。

5.6 参数估计

参数估计是概率统计学中经常使用的概念，包括置信区间、置信度、最大似然估计等参数，这些参数对于较多样本的考察具有很实用的价值。

5.6.1 unifit 函数——均匀分布的参数估计

【语法说明】

- [aht,bat] = unifit(X)：均匀分布参数的最大似然估计。
- [aht,bat,ACI,BCI] = unifit(X)：置信度为 95%的参数估计和置信区间。
- [aht,bat,ACI,BCI]=unifit(X, ALPHA)：返回水平α的参数估计和置信区间。

【功能介绍】 返回均匀分布参数的最大似然估计。

【实例 5.41】 求均匀分布向量 X=[1 3 5 7 9 11 13 15 17 19]的最大似然估计。

```
>> X=[1 3 5 7 9 11 13 15 17 19]

X =

     1     3     5     7     9    11    13    15    17    19

>> [aht,bat,ACI,BCI]=unifit(X, 0.04)                    %返回水平
α的参数估计和置信区间

ahat =

     1


bat =
```

```
    19


ACI =

  -431
     1


BCI =

    19
   451
```

【实例讲解】 ACI 和 BCI 是水平为 0.04 的置信区间。

5.6.2 normfit 函数——正态分布的参数估计

【语法说明】

- [muhat,sigmahat,muci,sigma] = normfit(X)：muhat,sigmahat 分别为正态分布的参数μ和σ的估计值，muci、sigma 分别为置信区间，其置信度为 $(1-\alpha)\times 100\%$。
- [muhat,sigmahat,muci,sigmaci] = normfit(X,alpha)：alpha 给出显著水平α，默认为 0.05，即置信度为 95%，其余参数同上。

【功能介绍】 求正态分布的参数估计。

【实例 5.42】 有两组正态随机数据，每组中有 100 个数值，其均值为 9，均方差为 1，求 96%的置信区间和参数估计值。

```
>> X = normrnd (9,1,100,2);      %产生两列正态随机数据
>> [mu,sigma,muci,sigmaci] = normfit(X,0.04)  %置信度为 96%的参数估计
mu =

   9.0567   8.8325

sigma =
```

```
    1.1340    1.0234

muci =

    8.8207    8.6195
    9.2927    9.0455


sigmaci =

    0.9896    0.8931
    1.3271    1.1977
```

【实例讲解】 muci、sigmaci 中各列分别为原随机数据各列估计值的置信区间，置信度为 96%。

5.6.3　binofit 函数——二项分布的参数估计

【语法说明】

- PHAT= binofit(X, N)：二项分布的概率的最大似然估计。
- [PHAT, PCI] = binofit(X,N)：置信度为 95%的参数估计和置信区间。
- [PHAT, PCI]= binofit (X, N, ALPHA)：返回水平为α的参数估计和置信区间。

【功能介绍】 计算二项分布的参数估计。

【实例 5.43】 进行二项分布的参数估计。

```
>> X=binornd(26,0.80)                    %产生二项分布的随机数

X =

    21

>> [PHAT, PCI]= binofit (X, 26, 0.95) %求概率的估计值和置信区间,置信度为 95%
```

```
PHAT =

    0.8077


PCI =

    0.7797    0.8273
```

【实例讲解】 得到的结果中 PHAT 为参数估计，PCI 为置信区间的上、下边界。

5.6.4 betafit 函数——计算β分布的参数估计

【语法说明】

- PHAT=betafit(X)：PHAT 为样本 X 的β分布的参数 a 和 b 的估计量。
- [PHAT,PCI]=betafit(X,ALPHA)：PCI 为样本 X 的β分布参数 a 和 b 的置信区间，是一个 2×2 矩阵，其第 1 列为参数 a 的置信下界和上界，第 2 列为参数 b 的置信下界和上界，ALPHA 为显著水平，（1−α）×100%为置信度。

【功能介绍】 β分布的参数 a 和 b 的最大似然估计值和置信区间。

【实例 5.44】 随机产生 100 个β分布数据，相应的分布参数真值为 5 和 6，求两个真值的最大似然估计值和置信度为 96%的置信区间。

```
>> X = betarnd (5,6,100,1);      %产生 100 个β分布的随机数
>> [PHAT,PCI] = betafit(X,0.04)%求置信度为 99%的置信区间和参数 a、b 的估计值
```

计算结果为：

```
PHAT =

    5.6206    5.9743
```

```
PCI =

    3.9572    4.0845
    7.2840    7.8642
```

【实例讲解】 估计值 5.6206 的置信区间是[3.9572 4.0845]，估计值 5.9743 的置信区间是[7.2840 7.8642]。

5.6.5 mle 函数——指定分布的参数估计

【语法说明】

- phat=mle ('dist',X)：X 为数据样本，返回用 dist 指定分布的最大似然估计值，dist 为分布函数名。
- [phat, pci]=mle('dist',X)：X 为数据样本，默认置信度为 95%。
- [phat, pci]=mle('dist',X,alpha)：X 为数据样本，置信度由 alpha 确定，alpha 为显著水平α，dist 为分布函数名。
- [phat, pci]=mle ('dist',X,alpha)：X 为数据样本，仅用于二项分布，dist 为分布函数名，pl 为试验次数。

【功能介绍】 利用 mel 函数进行参数估计。

【实例 5.45】 指定分布的参数估计。

```
>> X=binornd(20,0.8)          %产生二项分布的随机数

X =

    15

>> [p,pci]=mle('bino',X,0.04,20)     %求概率的估计值和置信区间,置信度为 95%

p =

    0.7500


pci =
```

```
    0.4984
    0.9182
```

【实例讲解】 [p,pci]=mle('bino',X,0.04,20) 利用 mle 进行参数估计，置信度为 1−0.04，20 表示数据的总数。

5.6.6 expfit 函数——指数分布的参数估计

【语法说明】

- muhat =expfit(X)：指数分布参数的最大似然估计。
- [muhat,muci] = expfit(X)：置信度为 95%的参数估计和置信区间。
- [muhat,muci] = expfit(X,alpha)：返回水平α的参数估计和置信区间。

【功能介绍】 返回指数分布参数的最大似然估计。

5.6.7 gamfit 函数——γ 分布参数的参数估计

【语法说明】

phat =gamfit(X)：γ分布参数的最大似然估计。

[phat,pci] = gamfit(X)：置信度为 95%的参数估计和置信区间。

[phat,pci] = gamfit(X,alpha)：返回最大似然估计值和水平α的置信区间。

【功能介绍】 求γ分布参数的参数估计。

5.6.8 weibfit 函数——韦伯分布的参数估计

【语法说明】

- phat = weibfit(X)：韦伯分布参数的最大似然估计。
- [phat,pci] = weibfit(X)：置信度为 95%的参数估计和置信区间。
- [phat,pci] = weibfit(X,alpha)：返回水平 α 的参数估计及其

区间估计。

【功能介绍】 求韦伯分布的参数估计。

5.6.9 poissfit 函数——泊松分布的估计值

【语法说明】

- Lambdahat=poissfit(X)：泊松分布的参数的最大似然估计。
- [Lambdahat, Lambdaci] = poissfit(X)：置信度为 95%的参数估计和置信区间。
- [Lambdahat, Lambdaci]= poissfit (X, ALPHA)：返回水平α的λ参数和置信区间。

【功能介绍】 求泊松分布的估计值。

【实例 5.46】 计算泊松分布的估计值。

```
>> X=binornd(26,0.80)  %产生随机分布函数

X =

    26

>> [Lambdahat, Lambdaci]= poissfit (X, 0.05)  %返回水
平α=95%的λ参数和置信区间

Lambdahat =

    26


Lambdaci =

   16.9841
   38.0960
```

【实例讲解】 [Lambdahat, Lambdaci]= poissfit (X, 0.05)求得置信度为 1−0.05 的信息，Lambdahat 表示λ参数，结果为 26，置信区间在 16.9841～38.0960 之间。

5.6.10 normfit 函数——正态分布的估计值

【语法说明】

- [muhat,sigmahat,muci,sigmaci] = normfit(X)：正态分布的最大似然估计，默认的置信度为 95%。
- [muhat,sigmahat,muci,sigmaci] = normfit(X, ALPHA)：返回水平α的期望、方差值和置信区间。

【功能介绍】 返回正态分布的估计值及置信区间等。

【实例 5.47】 计算正态分布的估计值。

```
>>  X=binornd(26,0.80)
X =
   20
>> [muhat,sigmahat,muci,sigmaci] = normfit(X, 0.05)
%返回水平α的期望、方差值和置信区间
muhat =
   20
sigmahat =

    0
muci =

  -Inf
   Inf
sigmaci =
    0
   Inf
```

【实例讲解】 [muhat,sigmahat,muci,sigmaci] = normfit(X, 0.05) 表示置信度为 1−0.05，得到的期望结果为 20，方差为 0，置信区间在负无穷与正无穷之间。

5.6.11 nlparci 函数——非线性模型的参数估计的置信区间

【语法说明】 ci = nlparci(beta,r,J)：返回置信度为 95%的置信区间，beta 为非线性最小二乘法估计的参数值，r 为残差，J 为 Jacobian

矩阵。nlparci 可以用 nlinfit 函数的输出作为其输入。

【功能介绍】 求非线性模型参数估计的置信区间。

【实例 5.48】 调用 MATLAB 中的数据 reaction.mat，并求其参数估计的置信区间。

```
>>load reaction
>>[beta,resids,J] = nlinfit(reactants,rate,'hougen',beta)
```

计算结果为：

```
beta =
   1.2526
   0.0628
   0.0400
   0.1124
   1.1914
resids =
   0.1321
  -0.1642
  -0.0909
   0.0310
   0.1142
   0.0498
  -0.0262
   0.3115
  -0.0292
   0.1096
   0.0716
  -0.1501
  -0.3026
J =
    6.8739  -90.6536  -57.8640   -1.9288    0.1614
    3.4454  -48.5357  -13.6240   -1.7030    0.3034
    5.3563  -41.2099  -26.3042  -10.5217    1.5095
    1.6950    0.1091    0.0186    0.0279    1.7913
    2.2967  -35.5658   -6.0537   -0.7567    0.2023
   11.8670  -89.5655 -170.1745   -8.9566    0.4400
    4.4973  -14.4262  -11.5409   -9.3770    2.5744
    4.1831  -41.7896  -16.8937   -5.7794    1.0082
   11.8286  -51.3721 -154.1164  -27.7410    1.5001
    9.1514  -25.5948  -76.7844  -30.7138    2.5790
    3.3373    0.0900    0.0720    0.1080    3.5269
```

```
    9.3663 -102.0611 -107.4327   -3.5811    0.2200
    4.7512  -24.4631  -16.3087  -10.3002    2.1141
>>ci = nlparci(beta,resids,J)
ci =
   -0.7467    3.2519
   -0.0377    0.1632
   -0.0312    0.1113
   -0.0609    0.2857
   -0.7381    3.1208
```

【实例讲解】 该例子是对数据文件的操作，由于数据文件中数据很多，所以结果以矩阵的形式存储。

5.6.12 nlpredci 函数——非线性模型置信区间预测

【语法说明】

- ypred = nlpredci(FUN,inputs,beta,r,J) ：ypred 为预测值，FUN 为用户提供的函数，beta 为给出的适当参数，r 为残差，J 为 Jacobian 矩阵，inputs 为非线性函数中的独立变量的矩阵值。
- [ypred,delta] = nlpredci(FUN,inputs,beta,r,J)：delta 为非线性最小二乘法估计的置信区间长度的一半，当 r 长度超过 beta 的长度并且 J 的列满秩时，置信区间的计算是有效的。[ypred-delta,ypred+delta] 为置信度为 95%的不同步置信区间。
- ypred = nlpredci(FUN,inputs,beta,r,J,alpha,'simopt','predopt')：控制置信区间的类型，置信度为 100(1–alpha)%。'simopt' = 'on'或'off'（默认值）分别表示同步或不同步置信区间。'predopt'='curve'（默认值）表示输入函数值的置信区间，'predopt'='observation'表示新响应值的置信区间。nlpredci 可以用 nlinfit 函数的输出作为其输入。

【功能介绍】 对非线性模型的置信区间进行预测。

5.6.13 lsnonneg 函数——非负最小二乘法

【语法说明】

- x = lsnonneg (A,b)：最小二乘法判断方程 A×x=b 的解，返

回在 x≥0 的条件下使得$\|Ax-b\|$最小的向量 x，其中 A 和 b 必须为实矩阵或向量。

- x = lsnonneg (A,b,tol)：A，b 同上，tol 为指定的误差。
- [x,w] = lsnonneg (A,b)：A，b 同上，当 x 中元素 $x_i = 0$ 时，$w_i \leqslant 0$，当 $x_i > 0$ 时 $w_i \cong 0$。
- [x,w] = lsnonneg (A,b,tol)：A，b 同上，tol 为指定的误差。

【功能介绍】 用最小二乘法判断方程的解。

【实例 5.49】 用最小二乘法判断方程的解。

```
>> A =[0.487 0.29;0.987 0.598;0.6 0.529;0.909 0.887]

A =
    0.4870    0.2900
    0.9870    0.5980
    0.6000    0.5290
    0.9090    0.8870
>> b=[0.57 0.371 0.07 0.234]'
b =
    0.5700
    0.3710
    0.0700
    0.2340
Warning: NNLS is obsolete and has been replaced by LSQNONNEG.
NNLS now calls LSQNONNEG which uses the following syntax:
[X,RESNORM,RESIDUAL,EXITFLAG,OUTPUT,LAMBDA]
=lsqnonneg(A,b,X0, Options) ;
Use OPTIMSET to define optimization options, or type
'edit nnls' to view the code used here.  NNLS will be
removed in the future; please use NNLS with the new syntax.
x =
         0
    0.6929
```

【实例讲解】 通过系数矩阵和常数向量，得到了方程的解，同

时给出警告信息。

5.6.14 lsqnonneg 函数——有非负限制的最小二乘法

【语法说明】

- x = lsqnonneg(C,d)：返回在 x≥0 的条件下使得 $\|Cx-d\|$ 最小的向量 x，其中 C 和 d 必须为实矩阵或向量。
- x = lsqnonneg(C,d,x0)：x0 为初始点，x0≥0。
- x = lsqnonneg(C,d,x0,options)：options 为指定的优化参数，参见 options 函数。
- [x,resnorm] = lsqnonneg(…)：resnorm 表示 norm(C*x−d).^2 的残差。
- [x,resnorm,residual] = lsqnonneg(…)：residual 表示 C*x−d 的残差。

【功能介绍】 返回有非负限制的最小向量。

【实例 5.50】 计算有非负限制的最小向量。

```
>> A =[0.0372 0.2869;0.6861 0.7071;0.6233 0.6245;0.6344
0.6170];
>> b=[0.8587 0.1781 0.0747 0.8405]';
>> [x,resnorm,residual] = lsqnonneg(A,b)
```

计算结果为：

```
x =
        0
   0.6929
resnorm =
   0.8315
residual =
   0.6599
  -0.3119
  -0.3580
   0.4130
```

【实例讲解】 在这种非负条件的限制下，得到的结果 x 一定是正数或 0，其他没有限制。

5.6.15 nlinfit 函数——高斯牛顿法的非线性最小二乘拟合

【语法说明】

- beta = nlinfit(X,y,FUN,beta0)：返回在 FUN 中描述的非线性函数的系数。FUN 为用户提供的函数，该函数返回已给初始参数估计值 β 和自变量 X 的 y 的预测值 $\hat{y}$。
- [beta,r,J] = nlinfit(X,y,FUN,beta0)：beta 为拟合系数，r 为残差，J 为 Jacobi 矩阵，beta0 为初始预测值。

【功能介绍】 求高斯牛顿法的非线性最小二乘拟合。

【实例 5.51】 调用 MATLAB 中的数据文件 reaction.mat，求其最小二乘拟和。

```
>>load reaction
>>betafit = nlinfit(reactants,rate,@hougen,beta)
```

计算结果为：

```
betafit =
    1.2526
    0.0628
    0.0400
    0.1124
    1.1914
```

【实例讲解】 利用 nlinfit 函数，同时加载 MATLAB 自身的数据文件，得到拟合的离散数据值。

5.6.16 nlintool 函数——非线性拟合

【语法说明】

- nlintool(x,y,FUN,beta0)：返回数据(x,y)的非线性曲线的预测图形，它用两条红色曲线预测全局置信区间。FUN 为用户提供的函数，beta0 为参数的初始预测值，置信度为 95%。
- nlintool(x,y,FUN,beta0,alpha)：置信度为(1−alpha)×100%。

【功能介绍】 对数据进行非线性拟和。

【实例 5.52】 调用 MATLAB 数据 reaction.mat，并对其进行非线性拟和运算。

```
>> load reaction
>> nlintool(reactants,rate,'hougen',beta)
```

对该数据文件进行非线性拟和得到的结果如图 5.14 所示。

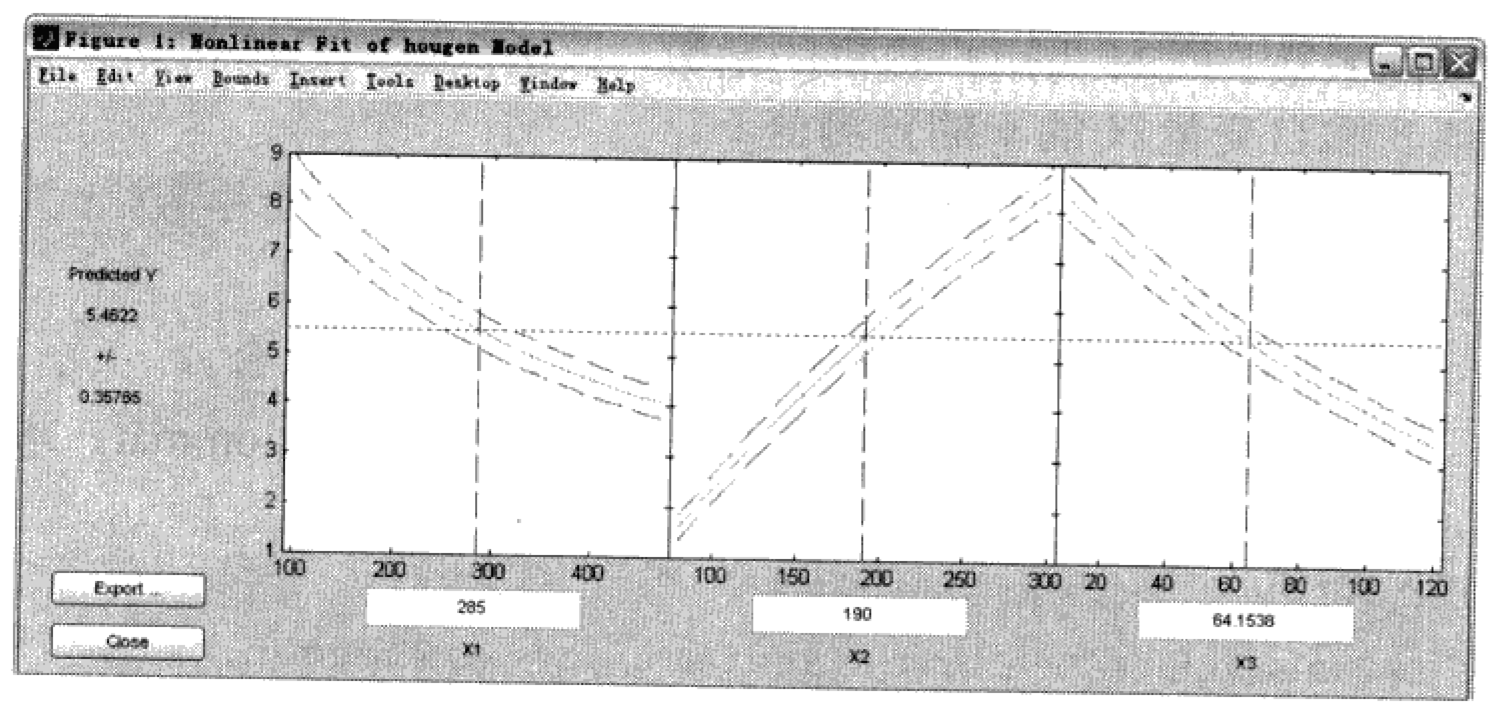


图 5.14 非线性拟和运算图形

【实例讲解】 用户可以在上面的界面中进行操作，查看结果。

5.6.17 betalike 函数——负β分布的对数似然函数

【语法说明】

- logL=betalike(params,data)：返回负β分布的对数似然函数，params 为向量[a, b]，是β分布的参数，data 为样本数据。
- [logL,info]=betalike(params,data)：返回 Fisher 逆信息矩阵 info。如果 params 中输入的参数是极大似然估计值，那么 info 的对角元素为相应参数的渐近方差。

【功能介绍】 求负β分布的对数似然函数。

【实例 5.53】 求随机产生的β分布数据的对数似然函数。

```
>> X = betarnd(5,5,100,1);
>> [logL,info] = betalike([1.4598,5.607],X)
```

计算结果为：

```
logL =

   92.6528

info =

    0.0511    0.0913
    0.0913    0.1921
```

【实例讲解】 首先用 betarnd 函数产生随机的负β分布函数，然后利用 betalike([1.4598,5.607],X)对向量 X 求对数似然函数。

5.6.18　gamlike 函数——负γ分布的对数似然估计

【语法说明】

- logL=gamlike(params,data)：返回由给定样本数据 data 确定的γ分布的参数为 params（即[a，b]）的负对数似然函数值。
- [logL,info]=gamlike(params,data)：返回 Fisher 逆信息矩阵 info。如果 params 中输入的参数是极大似然估计值，那么 info 的对角元素为相应参数的渐近方差。

【功能介绍】 求负γ分布的对数似然估计。

【实例 5.54】 负γ分布的对数似然估计。

```
>> X=gamrnd(4,8,100,1);
>> [logL,info]=gamlike([3.456, 3.789],X)
```

计算结果为：

```
logL =

  614.0603


info =

    0.0380   -0.0104
   -0.0104    0.0132
```

【实例讲解】 与上例相似，用 gamlike([3.456, 3.789],X)函数求

X 的负γ分布的对数似然估计。

5.6.19 normlike 函数——负正态分布的对数似然函数

【语法说明】

- logL=normlike(params,data)：返回由给定样本数据 data 确定的、负正态分布的、参数为 params（即[mu，sigma]）的对数似然函数值。
- [logL,info]=normlike(params,data)：返回 Fisher 逆信息矩阵 info。如果 params 中输入的参数是极大似然估计值，那么 info 的对角元素为相应参数的渐近方差。

【功能介绍】 求负正态分布的对数似然函数。

5.6.20 weiblike 函数——威布尔分布的对数似然函数

【背景知识】 威布尔分布的负对数似然函数定义：$-\log L=-\log\prod_{i=1}^{n}f(a,b\,|\,x_i)=-\sum_{i=1}^{n}\log f(a,b\,|\,x_i)$。

【语法说明】

- logL = weiblike(params,data)：返回由给定样本数据 data 确定的、威布尔分布的、参数为 params（即[a，b]）的对数似然函数值。
- [logL,info]=weiblike(params,data)：返回 Fisher 逆信息矩阵 info。如果 params 中输入的参数是极大似然估计值，那么 info 的对角元素为相应参数的渐近方差。

【功能介绍】 求威布尔分布的对数似然函数。

【实例 5.55】 求威布尔分布的对数似然函数。

```
>> X=weibrnd(0.3,0.9,100,1);
>> [logL,info]=weiblike([0.1234,0.8889],X)
```

计算的结果为：

```
logL =

  268.8655
```

```
info =

    0.0004   -0.0009
   -0.0009    0.0060
```

【实例讲解】 从上面的例子可以看出，威布尔分布的对数似然函数的计算结果。

5.7　假设检验

5.7.1　ttest 函数——*t* 检验法

【语法说明】

- h = ttest(x,m) ：x 为正态总体的样本，m 为均值μ_0，显著性水平为 0.05。
- h = ttest(x,m,alpha)：alpha 为给定显著性水平。
- [h,sig,ci] = ttest(x,m,alpha,tail)：sig 为观察值的概率，当 sig 为小概率时则对原假设提出质疑，ci 为真正均值μ 的 1−alpha 置信区间。

【功能介绍】 求取σ^2未知，单个正态总体的均值μ的假设检验。若 h=0，表示在显著性水平 alpha 下，不能拒绝原假设；若 h=1，表示在显著性水平 alpha 下，可以拒绝原假设。

【实例 5.56】 某种机械器件的长度 X（cm）服从正态分布，μ、σ^2 均未知。现测得 20 个器件的长度如下：159　280　238　101　212　224　379　179　264　222　362　168　250　149　260　485　170　190　226.5　230，问是否有理由认为该器件的平均长度大于 225（cm）？

解：未知σ^2，在水平α=0.05 下检验，假设：H_0：$\mu < \mu_0 = 225$，H_1：$\mu > 225$

```
>> X=[159    280    238    101    212    224    379    179
264   222   362   168   250 149   260   485   170   190
226.5  230]

X =

  Columns 1 through 8

  159.0000 280.0000 238.0000 101.0000 212.0000 224.0000
379.0000  179.0000

  Columns 9 through 16

  264.0000 222.0000 362.0000 168.0000 250.0000 149.0000
260.0000  485.0000

  Columns 17 through 20

  170.0000 190.0000 226.5000 230.0000

>> [h,sig,ci]=ttest(X,225,0.05,1)

h =

    0

sig =

   0.2688

ci =

  203.1978        Inf
```

【实例讲解】 均值 225 在置信区间内，通过结果可知：H=0 表示在水平$\alpha=0.05$下应该接受原假设 H_0，即认为元件的平均寿命不大于 225 小时。

5.7.2 ztest 函数——u 检验法

【语法说明】

- h = ztest(x,m,sigma)：x 为正态总体的样本，m 为均值μ_0，sigma 为标准差，显著性水平为 0.05（默认值）。
- h = ztest(x,m,sigma,alpha)：显著性水平为 alpha。
- [h,sig,ci,zval] = ztest(x,m,sigma,alpha,tail)：sig 为观察值的概率，当 sig 为小概率时则对原假设提出质疑，ci 为真正均值μ的 1−alpha 置信区间，zval 为统计量的值。

【功能介绍】 求已知σ^2，单个正态总体的均值μ的假设检验。若 h=0，表示在显著性水平 alpha 下，不能拒绝原假设；若 h=1，表示在显著性水平 alpha 下，可以拒绝原假设。

原假设：H_0：$\mu=\mu_0$=m。

【实例 5.57】 某车间用一台包装机包装食盐，食盐重量是一个随机变量，它服从正态分布。当机器正常时，其均值为 0.5kg，标准差为 0.02。某日开工后检验包装机是否正常，随机地抽取所包装的糖 10 袋，称得净重为（kg）

0.499, 0.507, 0.499 , 0.508, 0.524, 0.508, 0.491, 0.512, 0.519, 0.492

问机器是否正常？

解：总体μ和σ已知，那么这个题目是当σ^2为已知时，在水平$\alpha=0.05$下，根据样本值判断μ=0.5 还是$\mu\neq$0.5。为此提出假设：

令原假设：H_0：$\mu=\mu_0$=0.5

备择假设：H_1：$\mu\neq$0.5

```
>> X=[0.499, 0.507, 0.499 , 0.508, 0.524, 0.508, 0.491, 0.512, 0.519, 0.492]

X =

  Columns 1 through 8
```

```
    0.4990    0.5070    0.4990    0.5080    0.5240    0.5080
0.4910    0.5120

  Columns 9 through 10

    0.5190    0.4920

>>  [h,sig,ci,zval]=ztest(X,0.5,0.015,0.05,0)

h =

     0


sig =

    0.2136                    %样本观察值的概率


ci =

    0.4966    0.5152          %置信区间,均值 0.5 在此区间之内


zval =

    1.2438                    %统计量的值
```

【实例讲解】 通过结果可以知道 h=0，说明在水平$\alpha=0.05$ 的情况下，可接受原假设，即认为包装机工作是正常的。

5.7.3 signtest 函数——符号检验

【语法说明】

- p=signtest(X, Y, alpha)：X、Y 为两个总体的样本，长度必须相同，alpha 为显著性水平，P 为两个样本 X 和 Y 的中位数相等的概率，p 接近于 0 则可对原假设质疑。
- [p, h]=signtest(X, Y, alpha)：h 为检验结果：h=0 表示 X 与 Y

的中位数之差不显著，h=1 表示 X 与 Y 的中位数之差显著。

- [p,h,stats] = signtest(X,Y,alpha)：stats 中 sign 为符号统计量的值。

【功能介绍】 求两个总体中位数相等的假设检验。

【实例 5.58】 两个正态随机样本的中位数相等的假设检验。

```
>> X=normrnd(0,1,20,1);
>> Y=normrnd(0,2,20,1);
>> [p,h,stats]=signtest(X,Y,0.05)
p =
    0.2632
h =
     0
stats =
    sign: 7
```

【实例讲解】 结果表明：h=0 表示 X 与 Y 的中位数之差不显著。

5.7.4 ranksum 函数——秩和检验

【语法说明】

- p = ranksum(x,y,alpha)：x、y 为两个总体的样本，可以不等长，alpha 为显著性水平。
- [p,h] = ranksum(x,y,alpha)：h 为检验结果，h=0 表示 X 与 Y 的总体差别不显著；h=1 表示 X 与 Y 的总体差别显著。
- [p,h,stats] = ranksum(x,y,alpha)：stats 中包括 ranksum 为秩和统计量的值以及 zval 为过去计算 p 的正态统计量的值。

【功能介绍】 求两个总体一致性的检验。

【实例 5.59】 某商店为了确定向公司 A 或公司 B 购买某种商品，将 A 和 B 公司以往的各次进货的次品率进行比较，数据如下所示，设两样本独立。问两公司的商品的质量有无显著差异。设两公司的商品的次品的密度最多只差一个平移，取$\alpha = 0.05$。

A：7.0　3.5　9.6　8.1　6.2　5.1　10.4　4.0　2.0　10.5

B：5.7　3.2　4.1　11.0　9.7　6.9　3.6　4.8　5.6　8.4　10.1

5.5　12.3

解：设μ_A、μ_B分别为 A、B 两个公司的商品次品率总体的均值。则该问题为在水平$\alpha = 0.05$下检验假设：H_0：$\mu_A=\mu_B$，H_1：$\mu_A \neq \mu_B$。

```
>> A=[7.0 3.5 9.6 8.1 6.2 5.1 10.4 4.0 2.0 10.5];
>> B=[5.7 3.2 4.1 11.0 9.7 6.9 3.6 4.8 5.6 8.4 10.1 5.5 12.3];
>> [p,h,stats]=ranksum(A,B,0.05)
```

计算结果为：

```
p =
    0.8041
h =
     0
stats =
       zval: -0.2481
    ranksum: 116
```

【实例讲解】 通过结果可知：一方面，两样本总体均值相等的概率为 0.8041，不接近于 0；另一方面，H=0 也说明可以接受原假设 H_0，即认为两个公司的商品的质量无明显差异。

5.7.5　signrank 函数——符号秩检验

【语法说明】

- p = signrank(X,Y,alpha)：X、Y 为两个总体的样本，长度必须相同，alpha 为显著性水平，P 为两个样本 X 和 Y 的中位数相等的概率，p 接近于 0 则可对原假设质疑。
- [p,h] = signrank(X,Y,alpha)：h 为检验结果，h=0 表示 X 与 Y 的中位数之差不显著；h=1 表示 X 与 Y 的中位数之差显著。
- [p,h,stats] = signrank(x,y,alpha)：stats 中包括 signrank 为符号秩统计量的值以及 zval 为过去计算 p 的正态统计量的值。

【功能介绍】 求两个总体中位数相等的假设检验。

【实例 5.60】 两个正态随机样本的中位数相等的假设检验。

```
>> x=normrnd(0,1,20,1);
>> y=normrnd(0,2,20,1);
>> [p,h,stats]=signrank(x,y,0.05)
```

计算结果为：

```
p =
   0.3703
h =
     0
stats =
         zval: -0.8960
   signedrank: 81
```

【实例讲解】 通过结果可知：h=0 表示 X 与 Y 的中位数之差不显著。

5.7.6　ttest2 函数——两个正态总体均值差的检验（*t* 检验）

【语法说明】

- [h,sig,ci]=ttest2(X,Y)：X，Y 为两个正态总体的样本，显著性水平为 0.05。
- [h,sig,ci]=ttest2(X,Y,alpha)：alpha 为显著性水平。
- [h,sig,ci]=ttest2(X,Y,alpha,tail)：sig 为当原假设为真时得到观察值的概率，当 sig 为小概率时则对原假设提出质疑，ci 为真正均值μ的 1−alpha 置信区间。原假设：H_0：$\mu_1=\mu_2$，（μ_1 为 X 为期望值，μ_2 为 Y 的期望值）若 tail=0，表示备择假设：H_1：$\mu_1\neq\mu_2$（默认，双边检验）；tail=1，表示备择假设：H_1：$\mu_1>\mu_2$（单边检验）；tail=−1，表示备择假设：H_1：$\mu_1<\mu_2$（单边检验）。

【功能介绍】 两个正态总体方差未知但等方差时，比较两正态总体样本均值的假设检验。若 h=0，表示在显著性水平 alpha 下，不能拒绝原假设；若 h=1，表示在显著性水平 alpha 下，可以拒绝原假设。

【实例 5.61】 在平炉上进行一项试验以确定改变操作方法的建议是否会增加钢的产率，试验是在同一只平炉上进行的。每炼一炉钢时除操作方法外，其他条件都尽可能做到相同。先用标准方法炼一炉，然后用建议的新方法炼一炉，以后交替进行，各炼 10 炉，其产率分别如下。

（1）标准方法：78.1 72.4 76.2 74.3 77.4 78.4 76.0 75.5 76.7 77.3。

（2）新方法：79.1 81.0 77.3 79.1 80.0 79.1 79.1 77.3 80.2 82.1。

设这两个样本相互独立，且分别来自正态总体$N(\mu_1,\sigma^2)$和$N(\mu_2,\sigma^2)$，μ_1、μ_2、σ^2均未知。问建议的新操作方法能否提高产率？（取α=0.05）

解：两个总体方差不变时，在水平α=0.05 下检验假设：H_0：$\mu_1=\mu_2$，H_1：$\mu_1<\mu_2$。

```
>> X=[78.1 72.4 76.2 74.3 77.4 78.4 76.0 75.5 76.7 77.3];
>>Y=[79.1 81.0 77.3 79.1 80.0 79.1 79.1 77.3 80.2 82.1];
>> [h,sig,ci]=ttest2(X,Y,0.05,-1)
```

计算结果为：

```
h =
     1
sig =
  2.1759e-004      %说明两个总体均值相等的概率很小
ci =
       -Inf   -1.9083
```

【实例讲解】 H=1 表示在水平α=0.05 下，应该拒绝原假设，即认为建议的新操作方法提高了产率，因此，比原方法好。

5.7.7 jbtest 函数——正态分布的拟合优度测试

【语法说明】

- H = jbtest(X)：对输入向量 X 进行 Jarque-Bera 测试，显著性水平为 0.05。
- H = jbtest(X,alpha)：在水平 alpha 而非 5%下施行 Jarque-Bera 测试，alpha 在 0 和 1 之间。
- [H,P,JBSTAT,CV] = jbtest(X,alpha)：P 为接受假设的概率值，

P 越接近于 0，则可以拒绝是正态分布的原假设；JBSTAT 为测试统计量的值，CV 为是否拒绝原假设的临界值。

【功能介绍】 对正态分布的拟合优度进行测试。H 为测试结果，若 H=0，则可以认为 X 是服从正态分布的；若 X=1，则可以否定 X 服从正态分布。X 为大样本，对于小样本用 lillietest 函数。

【实例 5.62】 调用 MATLAB 中关于汽车重量的数据，测试该数据是否服从正态分布。

```
>> load carsmall
>> [h,p,j,cv]=jbtest(Weight)
h =
     1
p =
   0.0267
j =
   7.2448
cv =
   5.9915
```

【实例讲解】 p=2.67%表示应该拒绝服从正态分布的假设；h=1 也可否定服从正态分布；统计量的值 j = 7.2448 大于接受假设的临界值 cv =5.9915，因而拒绝假设（测试水平为 5%）。

5.7.8　kstest2 函数——两个样本具有相同的连续分布的假设检验

【语法说明】

- H = kstest2(X1,X2)：测试向量 X1 与 X2 是具有相同的连续分布，测试水平为 5%。
- H = kstest2(X1,X2,alpha)：alpha 为测试水平。
- [H,P,KSSTAT] = kstest(X,cdf,alpha)：与指定累积分布 cdf 相同的连续分布，P 为假设成立的概率，KSSTAT 为测试统计量的值。

【功能介绍】 求两个样本具有相同的连续分布的假设检验。原假设为具有相同连续分布，测试结果为 H，若 H=0，表示应接受原

假设；若 H=1，表示可以拒绝原假设。这是 Kolmogorov-Smirnov 测试方法。

【实例 5.63】 两样本具有相同连续分布的假设检验。

```
>> x=-1:1:5;
>> y=randn(20,1);
>> [h,p,k]=kstest2(x,y)
```

计算结果为：

```
h =
     1
p =
    0.0444
k =
    0.5643
```

【实例讲解】 h=1 表示可以认为向量 x 与 y 的分布不相同，相同的概率只有 4.4%。

5.7.9 kstest 函数——单个样本分布的 Kolmogorov-Smirnov 测试

【语法说明】

- H = kstest(X)：测试向量 X 是否服从标准正态分布，测试水平为 5%。
- H = kstest(X,cdf)：指定累积分布函数为 cdf 的测试(cdf=[ ] 时表示标准正态分布)，测试水平为 5%。
- H = kstest(X,cdf,alpha)：alpha 为指定测试水平。
- [H,P,KSSTAT,CV] = kstest(X,cdf,alpha)：P 为原假设成立的概率，KSSTAT 为测试统计量的值，CV 为是否接受假设的临界值。

【功能介绍】 测试单个样本。原假设为 X 服从标准正态分布。若 H=0 则不能拒绝原假设，H=1 则可以拒绝原假设。

【实例 5.64】 产生 100 个威布尔随机数，测试该随机数服从的分布。

```
>> x=weibrnd(1,2,100,1);
>> [H,p,ksstat,cv]=kstest(x,[x weibcdf(x,1,2)],0.05)
%测试是否服从威布尔分布
H =
     0
p =
    0.3022
ksstat =
    0.0959
cv =
    0.1340
>> [H,p,ksstat,cv]=kstest(x,[x expcdf(x,1)],0.05)%测
试是否服从指数分布
H =
     1
p =
    0.0073
ksstat =
    0.1653
cv =
    0.1340
```

说明 H=1 表明拒绝服从指数分布的假设。

```
>> [H,p,ksstat,cv]=kstest(x,[ ],0.05)       %测试是否服从标
准正态分布
H =
     1
p =
  3.1285e-026
ksstat =
    0.5380
cv =
    0.1340
```

说明 H=1 表明不服从标准正态分布。

【实例讲解】 从上面的例子中可以看出，使用 ktest 函数可以对假设条件进行判断。

5.8 图形绘制

5.8.1 lsline 函数——最小二乘拟合直线

【语法说明】

- lsline：最小二乘拟合直线。
- h = lsline：h 为直线的句柄。

【功能介绍】 绘制最小二乘拟和直线。

【实例 5.65】 求一列离散点参数 X = [1 2.4 3.6 5.9 8 11 12.3 13.7 16 18.9 20.8]的最小二乘拟合直线。

```
>> X = [1 2.4 3.6 5.9 8 11 12.3 13.7 16 18.9 20.8]

X =

  Columns 1 through 8

    1.0000    2.4000    3.6000    5.9000    8.0000   11.0000
12.3000   13.7000

  Columns 9 through 11

   16.0000   18.9000   20.8000

>>  plot(X,'+')
>> h=lsline

h =

  156.0018
```

上面代码得到的结果如图 5.15 所示。

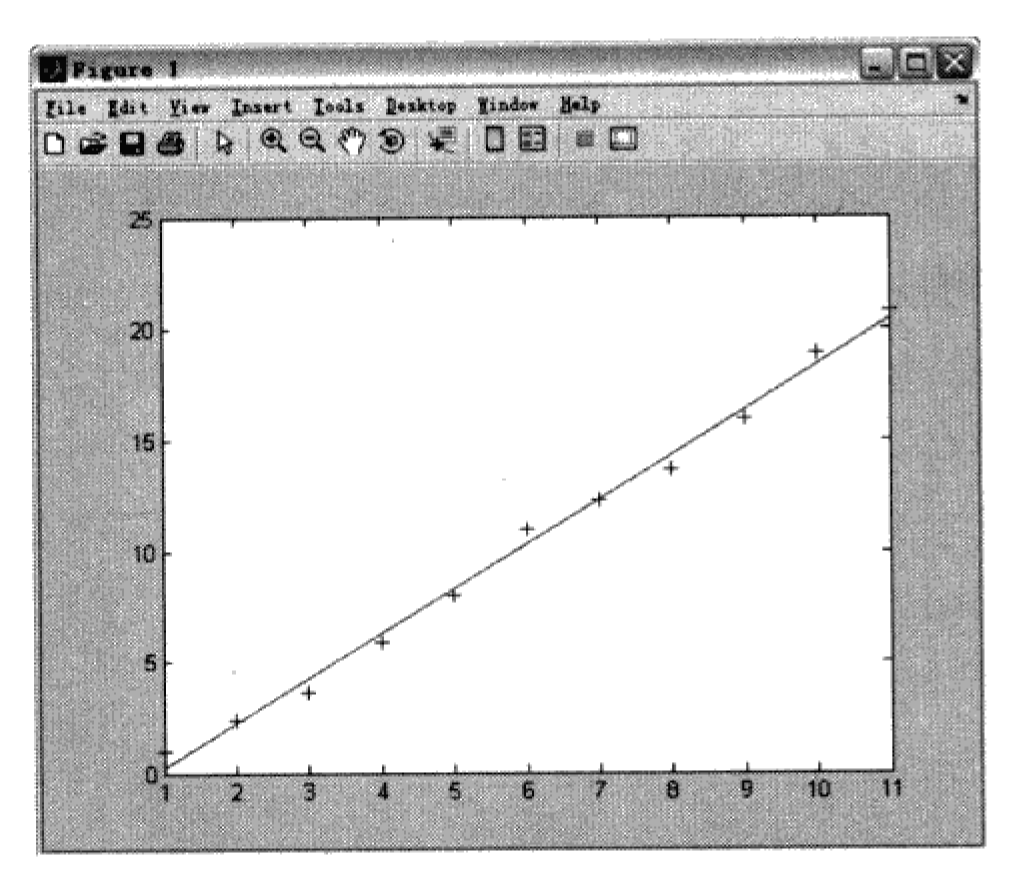


图 5.15 最小二乘拟合曲线

【实例讲解】 使用最小二乘法，图形中的离散数据点分别分散在拟合曲线四周。

5.8.2 normplot 函数——绘制正态分布概率图形

【语法说明】

- normplot(X)：若 X 为向量，则显示正态分布概率图形；若 X 为矩阵，则显示每一列的正态分布概率图形。
- h = normplot(X)：返回绘图直线的句柄。

样本数据在图中用“+”显示，如果数据来自正态分布，则图形显示为直线，而其他分布可能在图中产生弯曲。

【功能介绍】 绘制正态分布概率图形。

【实例 5.66】 绘制正态分布概率图。

```
>> X=normrnd(0,1,60,1);
>> normplot(X)
```

得到的结果如图 5.16 所示。

【实例讲解】 从结果可以看出，图形的坐标轴并不是线性的。

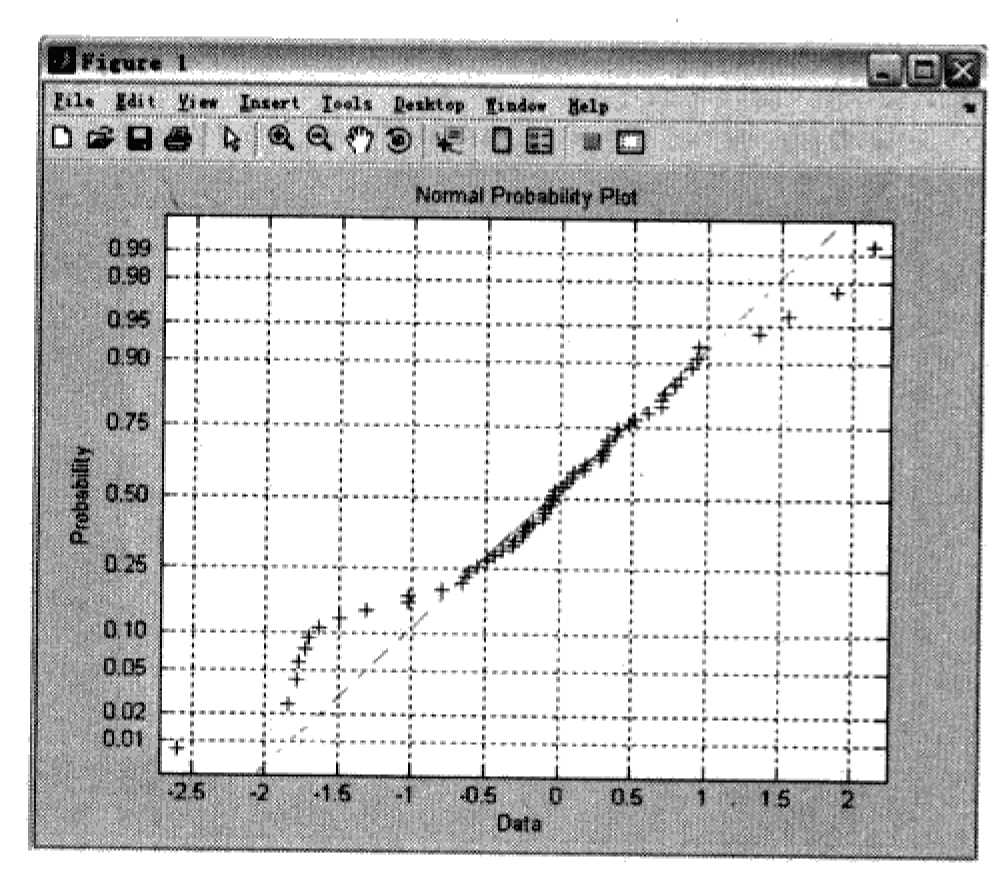


图 5.16 正态分布概率图

5.8.3 tabulate 函数——正整数的频率表显示

【语法说明】 table = tabulate(X)：X 为正整数构成的向量，返回 3 列。第 1 列中包含 X 的值第 2 列为这些值的个数，第 3 列为这些值的频率。

【功能介绍】 列出正整数的频率表。

【实例 5.67】 求向量 A=[1 2 2 5 6 3 8]的正整数频率表。

```
>> A=[1 2 2 5 6 3 8]
A =
     1     2     2     5     6     3     8
>> tabulate(A)
  Value    Count   Percent
      1        1     14.29%
      2        2     28.57%
      3        1     14.29%
      4        0      0.00%
      5        1     14.29%
```

```
        6        1     14.29%
        7        0      0.00%
        8        1     14.29%
```

【实例讲解】 正整数的频率分布表在统计分数等情况下有广泛的应用。

5.8.4 capaplot 函数——样本的概率图形

【语法说明】 p = capaplot(data,specs)：data 为所给样本数据，specs 指定范围，p 表示在指定范围内的概率。

【功能介绍】 该函数返回来自于估计分布的随机变量落在指定范围内的概率。

【实例 5.68】 绘制样本的概率图形。

```
>> data=normrnd (0,1,30,1);
>> p=capaplot(data,[-2,2])
p =
      0.9199
```

得到的结果如图 5.17 所示。

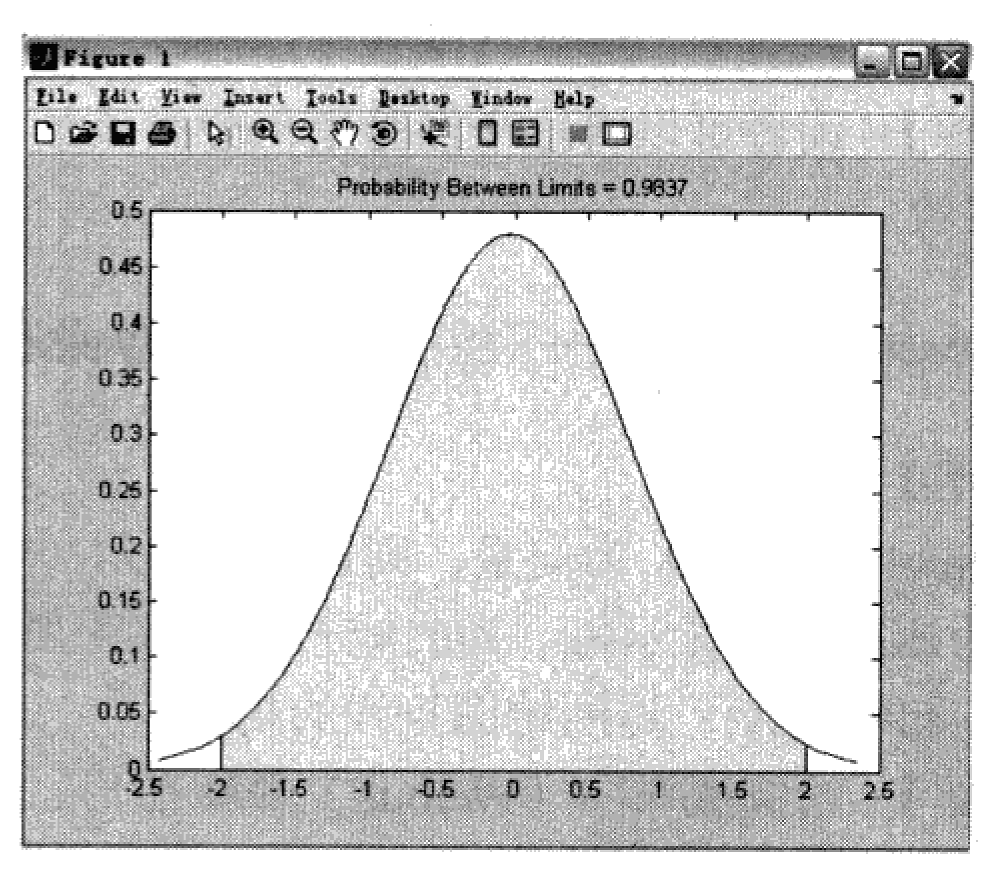


图 5.17　样本的概率图形

【实例讲解】 上面的图形形象地表示出样本的概率数值。

5.8.5 cdfplot 函数——经验累积分布函数图形

【语法说明】

- cdfplot(X)：作样本 X（向量）的累积分布函数图形。
- h = cdfplot(X)：h 表示曲线的环柄。
- [h,stats] = cdfplot(X)：stats 表示样本的一些特征。

【功能介绍】 显示经验累积分布函数图形。

【实例 5.69】 绘制经验累积分布函数图形。

```
>> X=normrnd (0,1,60,3);
>> [h,stats]=cdfplot(X)
h =
   3.0013
stats =
      min: -1.8740       %样本最小值
      max: 1.6924        %最大值
     mean: 0.0565        %平均值
   median: 0.1032        %中间值
      std: 0.7559        %样本标准差
```

【实例讲解】 读者可以将这个经验累积分布函数和正态分布的累积函数相比较。

5.8.6 weibplot 函数——绘制威布尔（Weibull）概率图形

【语法说明】

- weibplot(X)：若 X 为向量，则显示威布尔（Weibull）概率图形；若 X 为矩阵，则显示每一列的威布尔概率图形。
- h = weibplot(X)：返回绘图直线的柄。

【功能介绍】 绘制威布尔（Weibull）概率图形。

【实例 5.70】 绘制威布尔概率图形。

```
>> X= weibrnd(2.3,2.9,60,1);
>> weibplot(X)
```

得到的结果如图 5.18 所示。

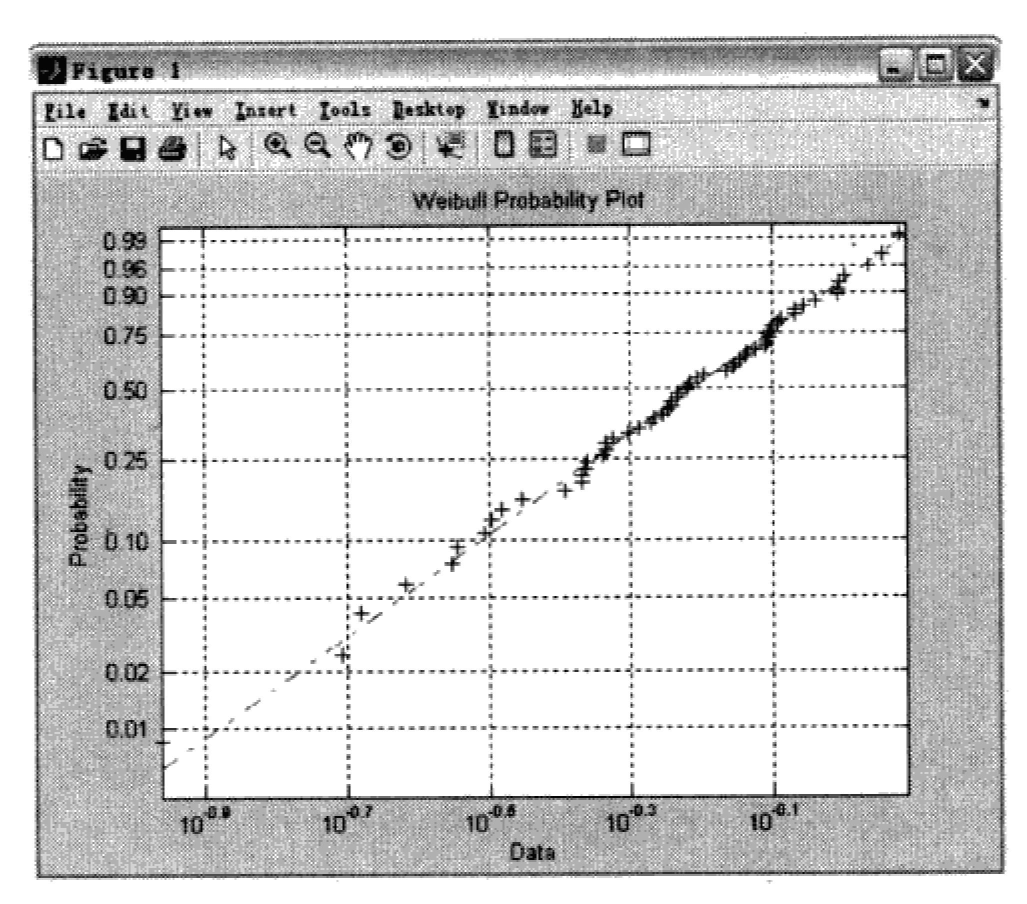


图 5.18 威布尔（Weibull）概率图形

【实例讲解】 关于威布尔（Weibull）概率的详细知识，请用户查阅相应的书籍。

5.8.7 histfit 函数——带有正态密度曲线的直方图

【语法说明】

- histfit(data)：data 为向量，返回直方图和正态曲线。
- histfit(data,nbins)：nbins 指定 bar 的个数，缺省时为 data 中数据个数的平方根。

【功能介绍】 绘制带有正态密度曲线的直方图。

【实例 5.71】 绘制正态密度的直方图。

```
>>r = normrnd (10,1,100,1);
>>histfit(r)
```

得到的结果如图 5.19 所示。

【实例讲解】 读者可以修改函数的参数，查看直方图和正态密度曲线之间的关系。

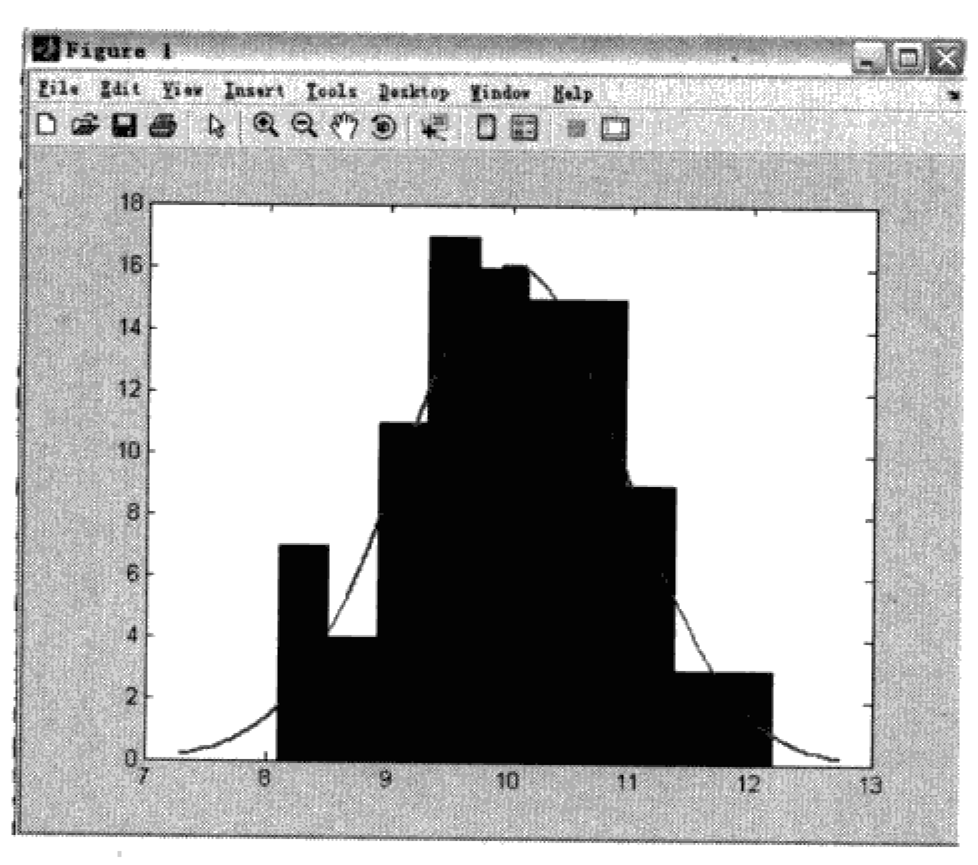


图 5.19　带有正态密度曲线的直方图

5.8.8　boxplot 函数——样本数据的盒图

【语法说明】

- boxplot(X)：产生矩阵 X 的每一列的盒图和“须”图，“须”是从盒的尾部延伸出来，并表示盒外数据长度的线，如果“须”的外面没有数据，则在“须”的底部有一个点。
- boxplot(X,notch)：当 notch=1 时，产生一凹盒图，notch=0 时产生一矩箱图。
- boxplot(X,notch,'sym')：sym 表示图形符号，默认值为“+”。
- boxplot(X,notch,'sym',vert)：当 vert=0 时，生成水平盒图，当 vert=1 时，生成竖直盒图（默认值 vert=1）。
- boxplot(X,notch,'sym',vert,whis)：whis 定义“须”图的长度，当默认值为 1.5，若 whis=0 则 boxplot 函数通过绘制 sym 符号图来显示盒外的所有数据值。

【功能介绍】 绘制样本数据的盒图。

【实例 5.72】 绘制样本数据的盒图。

```
>>X1 = normrnd(6,2,200,1);
>>X2 = normrnd(8,2,200,1);
>>X= [X1 X2];
>> boxplot(X,1,'g+',1,0)
```

得到的结果如图 5.20 所示。

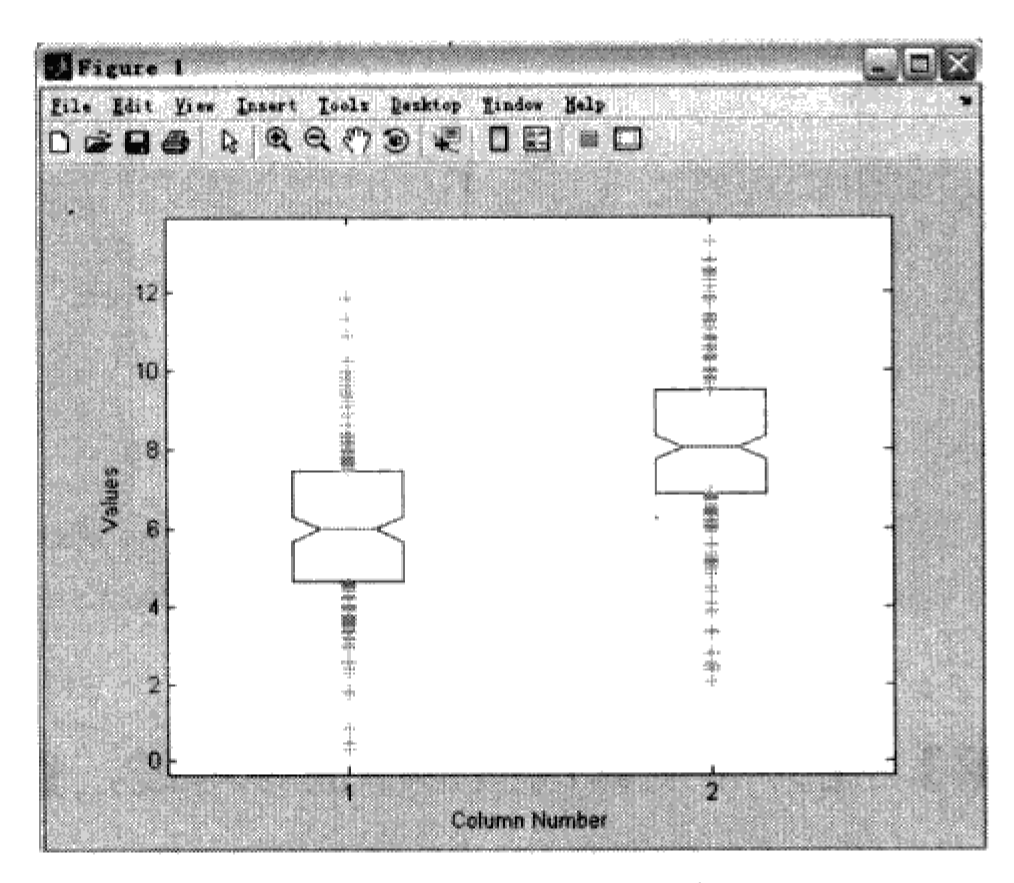


图 5.20　样本数据的盒图

【实例讲解】 读者可以修改参数数值，绘制水平方向的盒图。

5.8.9　refline 函数——给当前图形加一条参考线

【语法说明】

- refline(slope,intercept)：slope 表示直线斜率，intercept 表示截距。
- refline(slope)：slope=[a b]，图中加一条直线 $y=b+ax$。

【功能介绍】 给当前图形加一条参考线。

【实例 5.73】 给当前的离散数据阵列 y = [3.2 2.6 3.1 3.4 2.4 2.9 3.0 3.3 3.2 2.1 2.6]加一条参考线。

```
>>y = [3.2 2.6 3.1 3.4 2.4 2.9 3.0 3.3 3.2 2.1 2.6]';
>>plot(y,'--')
>>refline(0,3)
```

得到的结果如图 5.21 所示。

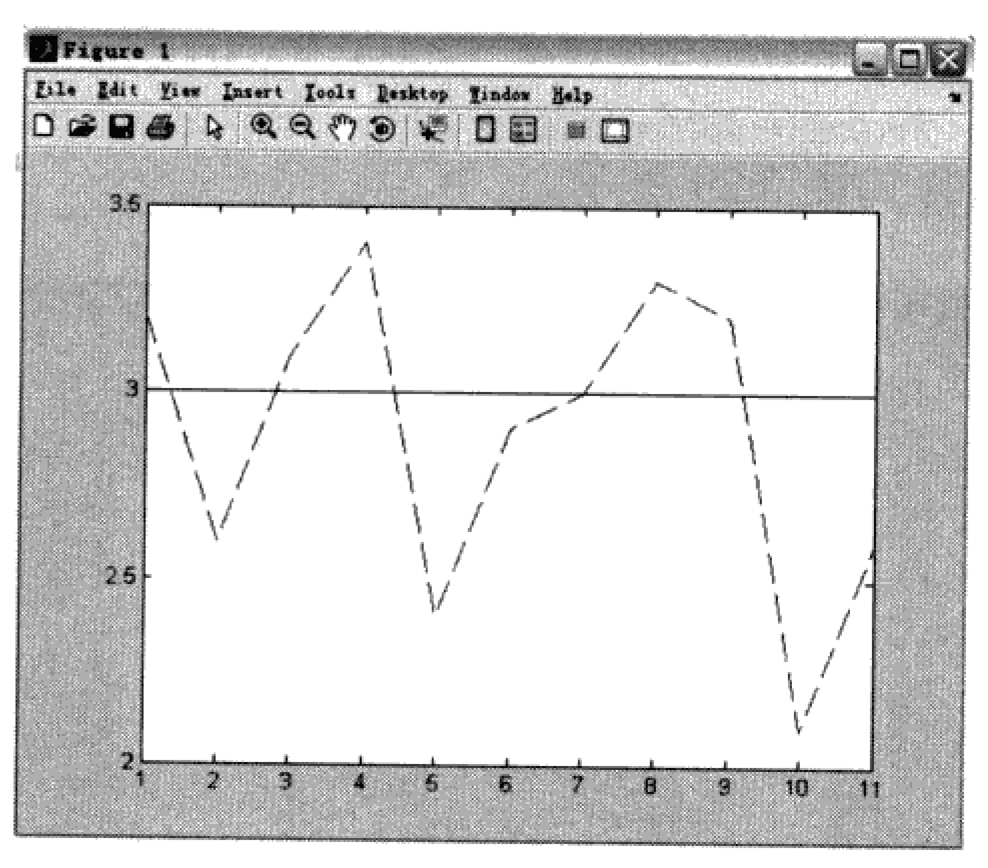


图 5.21　参考线绘制

【实例讲解】 用户可从修改图形中的参考线，查看修改后的图表。

5.8.10　refcurve 函数——在当前图形中加入一条多项式曲线

【语法说明】 h = refcurve(p)：在图中加入一条多项式曲线，h 为曲线的环柄，p 为多项式系数向量，p=[p1,p2, p3,…,pn]，其中 p1 为最高幂项系数。

【功能介绍】 在当前图形中加入一条多项式曲线。

5.8.11　normspec 函数——在指定的界线之间画正态密度曲线

【语法说明】 p = normspec(specs,mu,sigma)：specs 指定界线，mu、sigma 为正态分布的参数 p 的样本落在上、下界之间的概率。

【功能介绍】 在指定的界线之间画正态密度曲线。

【实例 5.74】 在[10，+∞]上画正态密度曲线。

```
>>normspec([10 Inf],11.5,1.25)
```

得到的结果如图 5.22 所示。

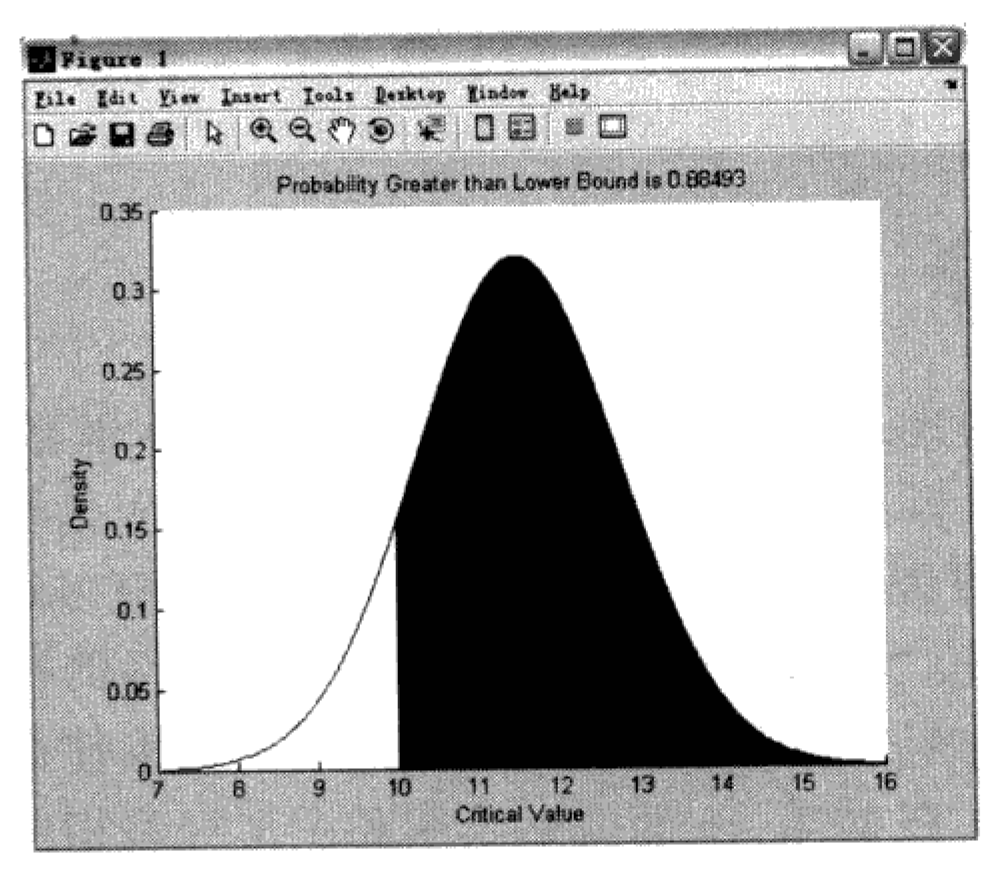
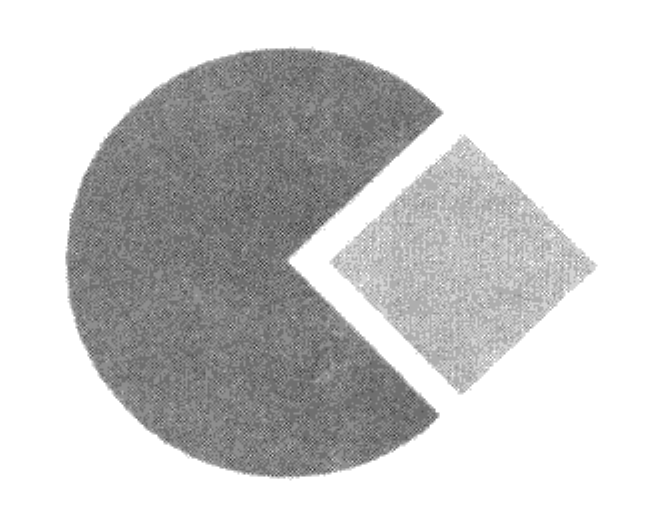


图 5.22 正态密度曲线

【实例讲解】 读者可以修改图形中的参考线，查看修改后的图表。



第6章　绘图与图形处理

在众多的表现手法中，数据图形是最能体现数据内在本质的，而且从中容易发现数据的内在联系，甚至变化发展的趋势。MATLAB 可以表达出数据的二维、三维，甚至更多维的图形。通过图形的线型、立面、色彩、光线、视角等属性的控制，可把数据的内在特征充分表现。这种直观、简洁的特性使得绘图函数在 MATLAB 中广泛应用。下面就对绘图和图形处理相关的函数作初步的讲解。

6.1　二维图形

作为一个功能强大的工具软件，MATLAB 具有很强的图形处理功能，提供了大量的二维、三维图形函数。由于系统采用面向对象的技术和丰富的矩阵运算，因此在图形处理方面既方便又高效。本节主要讲解 MATLAB 二维图形的绘制功能。

6.1.1　plot 函数——基本平面图形函数

【语法说明】

- plot(X，Y)：当 X、Y 均为实数向量，且为同维向量，X=[x(i)]，Y=[y(i)]时，则 plot(X,Y)先描出点(x(i)，y(i))，然后用直线依次相连；若 X，Y 为复数向量，则不考虑虚数部分。

- plot(Y)：若 Y 为实数向量，Y 的维数为 n，则 plot(Y)等价于 plot(X,Y)，其中 x=1：n；若 Y 为实数矩阵，则把 Y 按列的方向分解成几个列向量，而 y 的行数为 n，则 plot(Y)等价于 plot(X,Y)其中 x=[1;2;…;n]。
- plot(X1,Y1,X2,Y2,…)：Xi 与 Yi 成对出现，plot(X1,Y1,X2,Y2,…)将分别按顺序取两数据 Xi 与 Yi 进行画图。若其中仅仅有 Xi 或 Yi 是矩阵，其余的为向量，向量维数与矩阵的维数匹配，则按匹配的方向来分解矩阵，再分别将配对的向量画出。

【功能介绍】 绘制线性二维图。在线条多于一条时，若用户没有指定使用颜色，则 plot 循环使用由当前坐标轴颜色顺序属性定义的颜色，以区别不同的线条。

【实例 6.1】 在区间 $0 \leqslant x \leqslant 2$ 内，绘制正弦曲线 $y=\sin(x)$。

```
x=0:pi/100:2*pi;
y=sin(x);
plot(x,y)
```

结果得到的是正弦函数曲线，如图 6.1 所示。

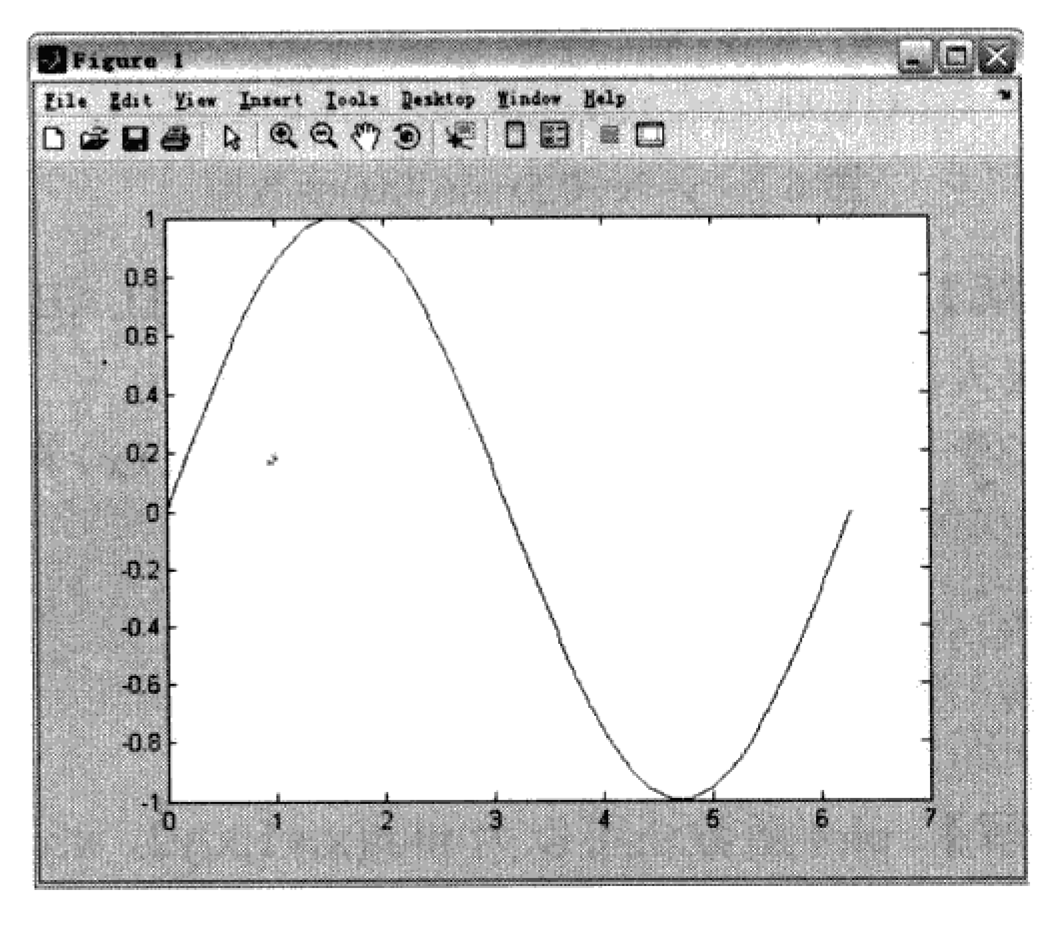


图 6.1　正弦函数

【实例讲解】 在用户没有设定图表的各种属性时，MATLAB 将使用默认属性绘制图表。

【实例 6.2】 在 $0 \leqslant x \leqslant 2\pi$ 区间内，绘制曲线 y=2e−0.5$x$cos(4πx)。

```
x=0:pi/100:2*pi;
y=2*exp(-0.5*x).*cos(4*pi*x);
plot(x,y)
```

通过曲线的函数关系式，可以看出，这是一个振幅逐渐衰减的函数。由这个例子得到的曲线图如图 6.2 所示。

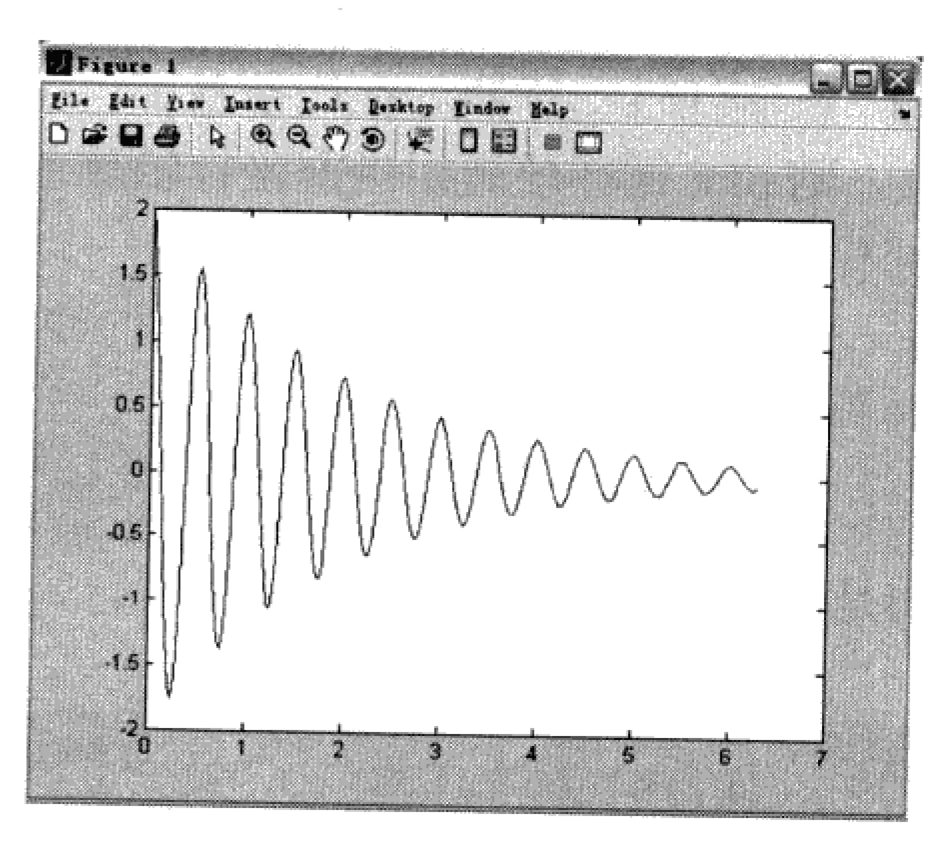


图 6.2　y=2e−0.5$x$cos(4πx)函数图

【实例讲解】 从上面的例子中可以看出，MATLAB 在绘制超越函数时，有很大的优势。

【实例 6.3】 同时绘制正、余弦两条曲线 y1=sin（x）和 y2=cos（x）。

```
x=0:pi/100:2*pi;
y1=sin(x);
y2=cos(x);
plot(x,y1,x,y2)
```

上例得到图形如图 6.3 所示。

【实例讲解】 plot 函数还可以为 plot(x,y1,x,y2，x,y3，…)形式，其功能是以公共向量 x 为 X 轴，分别以 y1，y2，y3，…为 Y 轴，在同一幅图内绘制出多条曲线。

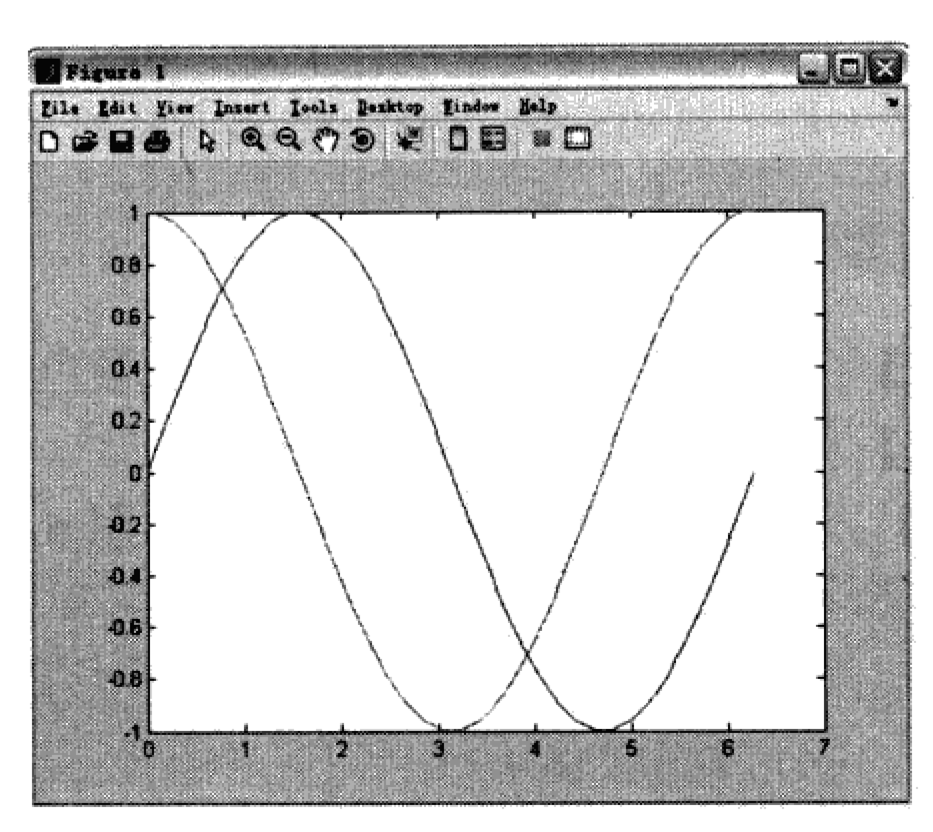


图 6.3　正余弦趋向图

【实例 6.4】 在一个图中绘制 3 条曲线。

```
>>t = 0:pi/20:2*pi;
>>plot(t,t.*cos(t),'-.r*')
>>hold on
>>plot(exp(t/100).*sin(t-pi/2),'--mo')
>>plot(sin(t-pi),':bs')
>>hold off
```

上例得到图形如图 6.4 所示。

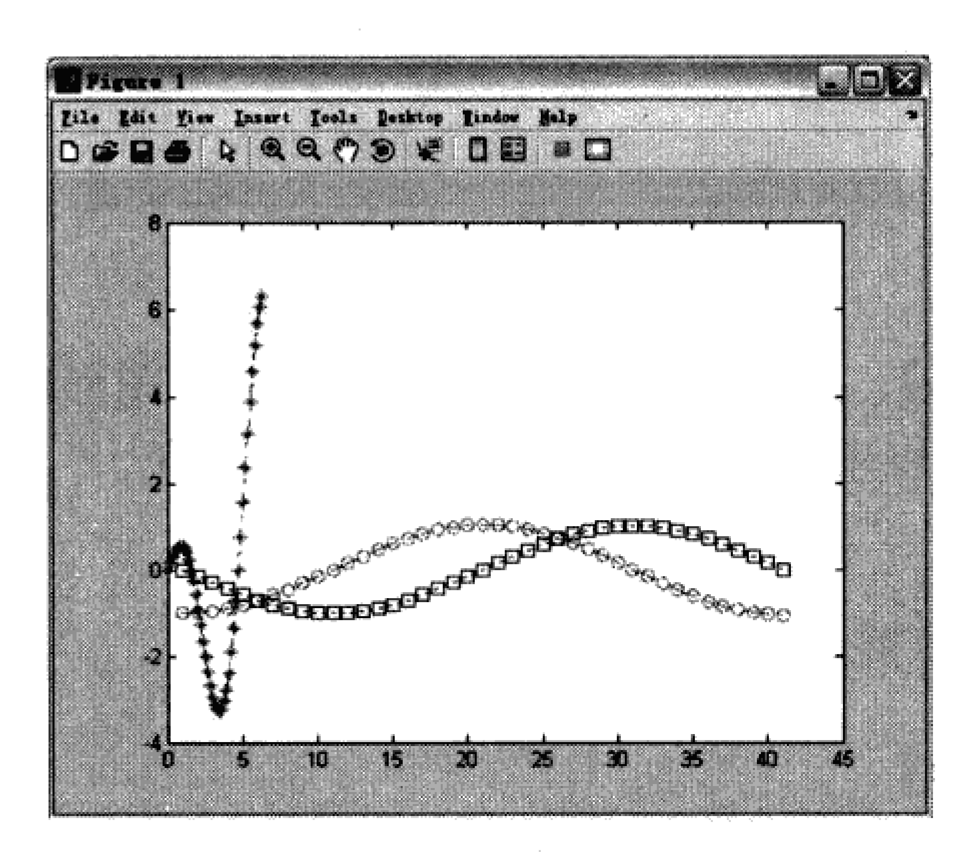


图 6.4　3 条曲线

【实例讲解】 用来指定自变量的定义域，plot 函数中的 3 个参数分别指定了横坐标、纵坐标和绘制曲线的点的形状。

6.1.2 线型与颜色

【语法说明】 plot(x,y1,'cs',...)：其中 c 表示颜色，s 表示线型。

【功能介绍】 设定绘图的颜色和线性。

【实例 6.5】 用不同线型和颜色重新绘制如图 6.3 所示图形。

```
x=0:pi/100:2*pi;
y1=sin(x);
y2=cos(x);
plot(x,y1,'go',x,y2,'b-.')
```

得到的结果如图 6.5 所示。

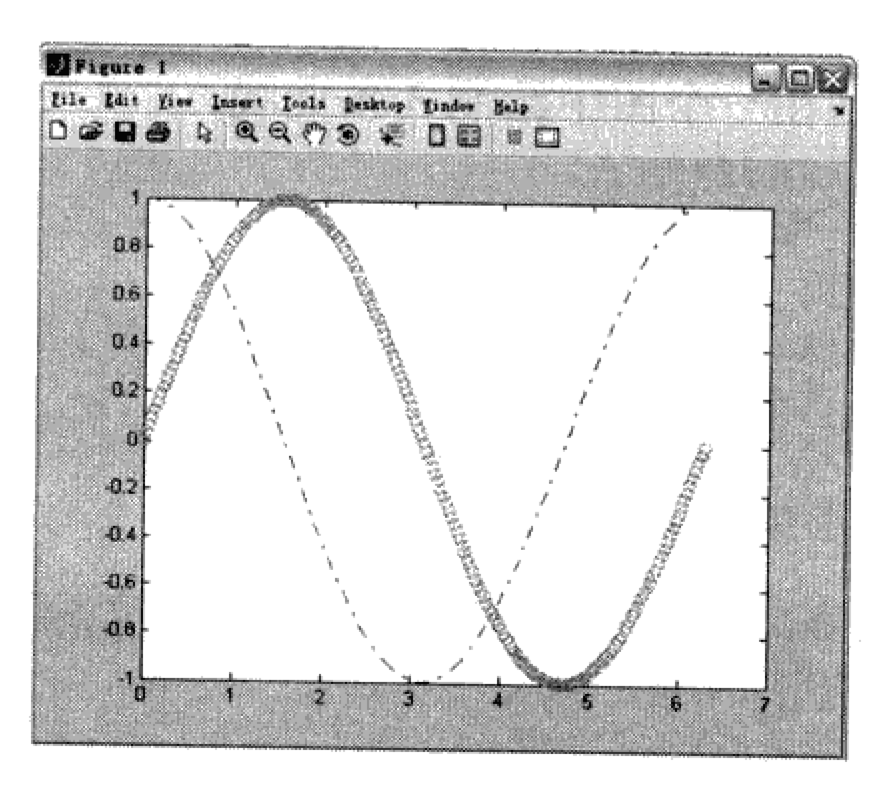


图 6.5 利用线型和颜色

【实例讲解】 其中参数“go”和“b-”表示图形的颜色和线型。g 表示绿色，o 表示图形线型为圆圈；b 表示蓝色，“-”.表示图形线型为点划线。

6.1.3 图形标记

【语法说明】

- title('加图形标题')。

- xlabel('加 X 轴标记')。
- ylabel('加 Y 轴标记')。

【功能介绍】 在绘制图形的同时，可以对图形加上一些说明，如图形名称、图形某一部分的含义、坐标说明等，将这些操作称为添加图形标记。

【实例 6.6】 为正弦函数作标记。

```
>>x=0:pi/100:2*pi;
>>y1=sin(x);
>>plot(x,y1,'go')
>>title('正弦曲线');
>> xlabel('横坐标 X');
>> ylabel('纵坐标 Y');
```

作标记后的图形如图 6.6 所示。

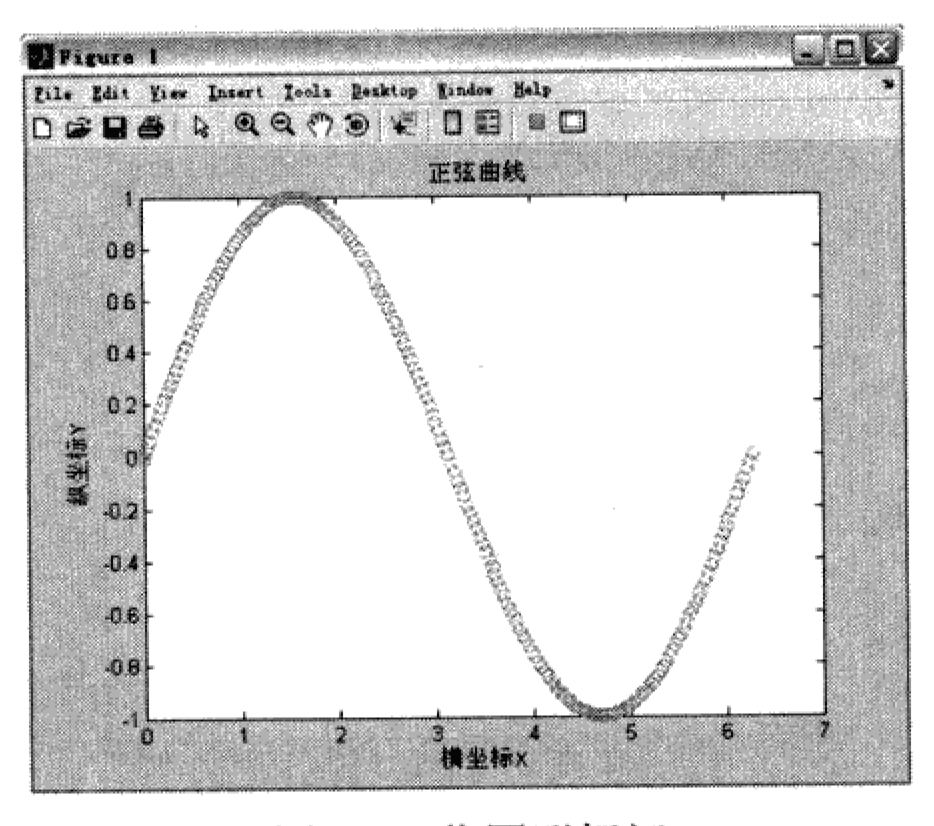


图 6.6 作图形标记

【实例讲解】 使用图形标记，可以为图表提供更活的信息。

6.1.4 设定坐标轴

【语法说明】

- axis([xmin xmax ymin ymax])：设定最大和最小值。
- axis （'auto'）：将坐标系统返回到自动默认状态。

- axis （'square'）：将当前图形设置为方形。
- axis （'equal'）：两个坐标因子设成相等。
- axis （'off'）：关闭坐标系统。
- axis （'on'）：显示坐标系统。

【功能介绍】 用户若对坐标系统不满意，可利用 axis 函数对其重新设定。

【实例 6.7】 在坐标范围 $0 \leqslant X \leqslant 2\pi$，$-2 \leqslant Y \leqslant 2$ 内重新绘制正弦曲线。

```
x=linspace(0,2*pi,60);    %生成含有 60 个数据元素的向量 X
y=sin(x);
plot(x,y);
axis ([0 2*pi -2 2]);        %设定坐标轴范围
```

得到的结果如图 6.7 所示。

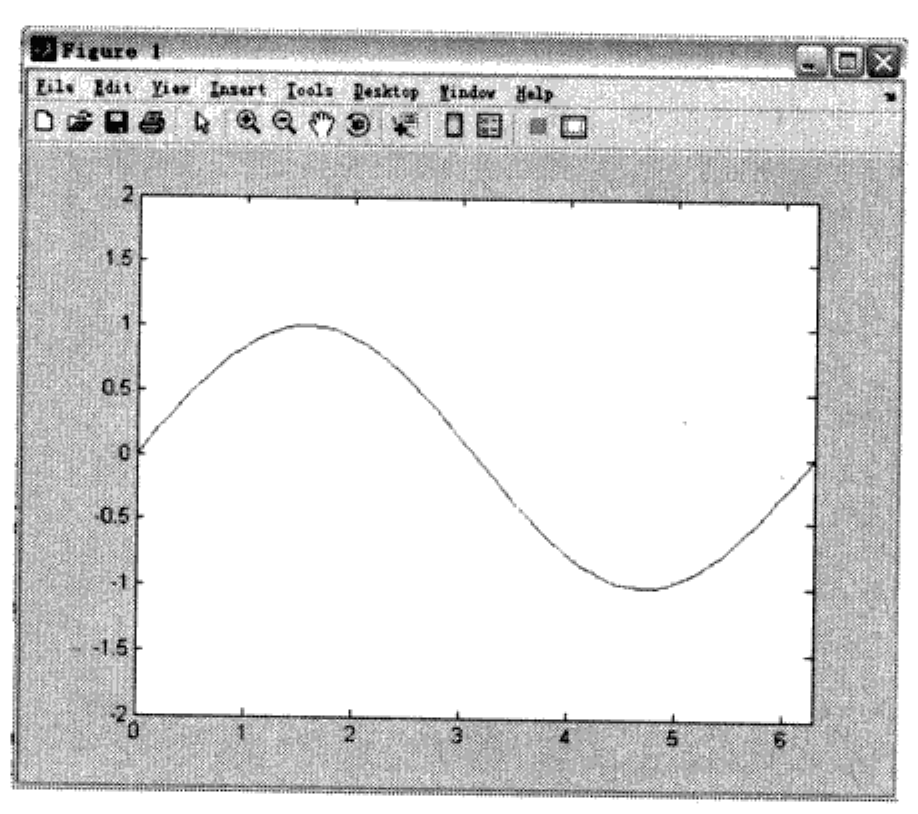


图 6.7 标轴重新设定后的正弦函数

【实例讲解】 本例中将横坐标轴的范围设为 0～2*pi，纵坐标的范围设为-2～2。

【实例 6.8】 在坐标范围设定为方形重新绘制正弦曲线。

```
x=linspace(0,2*pi,60);    %生成含有 60 个数据元素的向量 X
y=sin(x);
plot(x,y);
axis ('square')    %将当前图形设置为方形
```

得到的结果如图 6.8 所示。

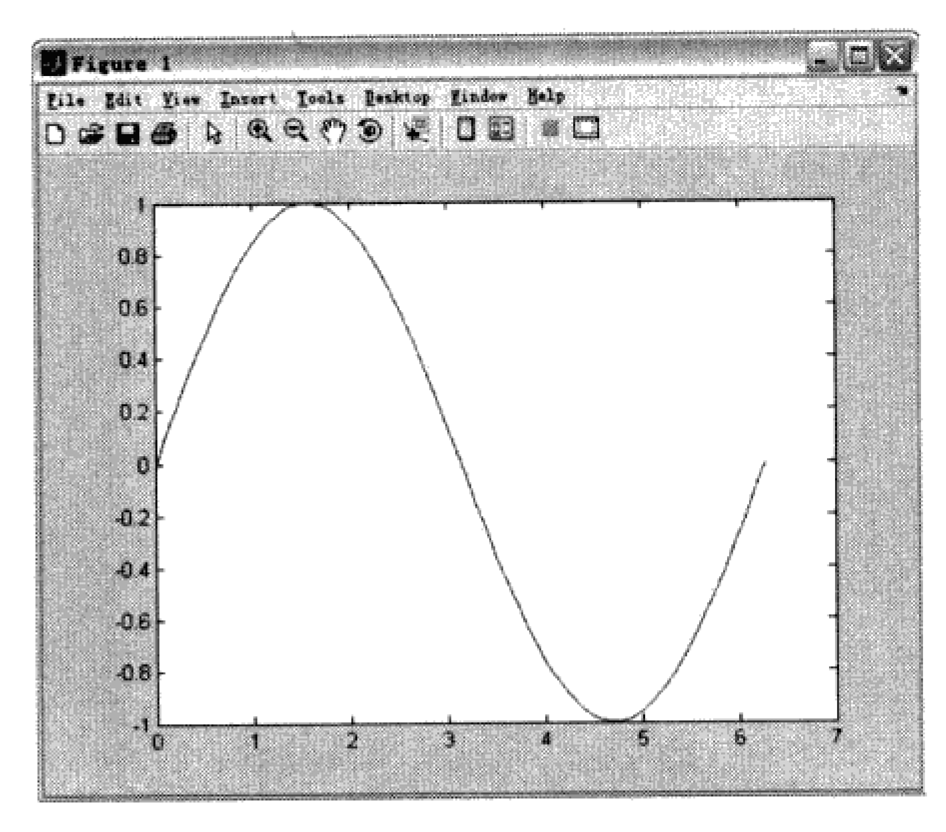


图 6.8 坐标轴设定成正方形后的正弦函数

【实例讲解】 通过 aixs 函数可以设定坐标轴的属性，更加便利地显示图形。

6.1.5 legend 函数——加图例

【语法说明】 legend('图例说明','图例说明')。

【功能介绍】 该函数把图例放置在图形空白处，用户可以通过鼠标移动图例，将其放到希望的位置。

【实例 6.9】 为正弦、余弦曲线增加图例。

```
x=0:pi/100:2*pi;
y1=sin(x);
y2=cos(x);
plot(x,y1,x,y2, '--');
legend('sin(x)','cos(x)');
```

得到的结果如图 6.9 所示。

【实例讲解】 在下面的图形中，可以看到，在两条曲线的右上方的小方框中，分别以曲线的线型表示是正弦还是余弦。也就是上面命令中，最后一行所说明的内容。

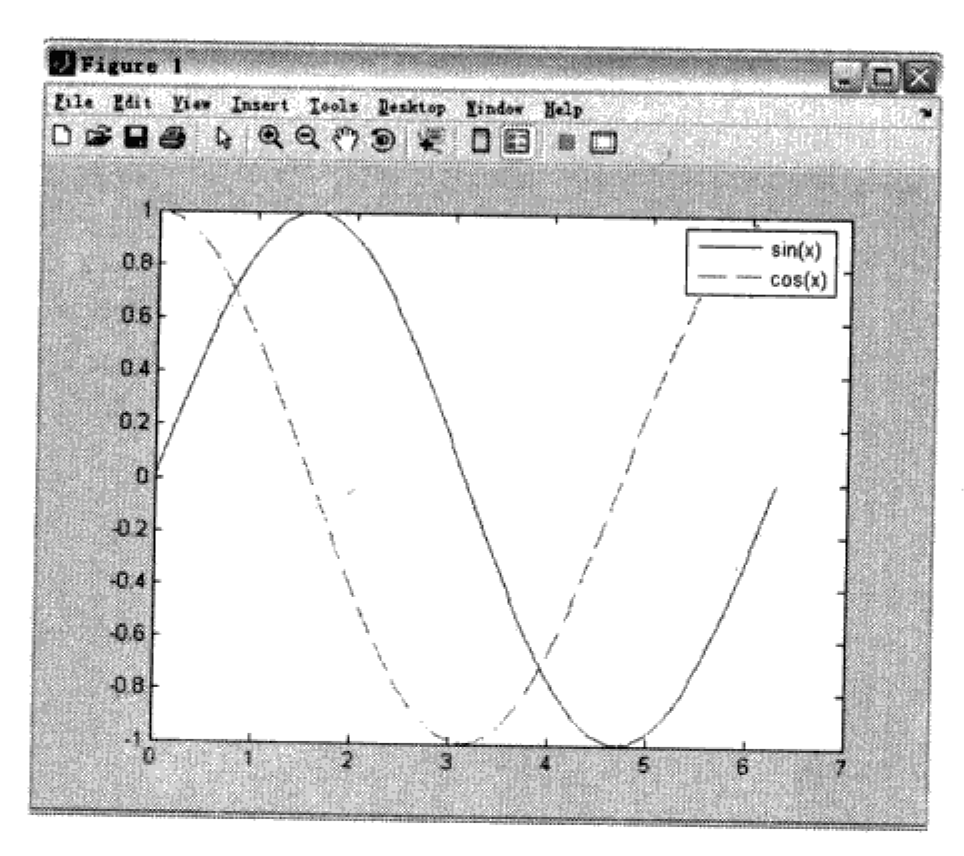


图 6.9　为曲线加图例

6.1.6　text 函数——添加字符串

【语法说明】

- text(x,y,'string')：在图形中指定的位置（x,y）上显示字符串 string。
- text(x,y,z,'string')：在三维图形空间中的指定位置（x,y,z）上显示字符串 string。
- text(x,y,z,'string'.'属性名称', '属性值')：对引号中的文字 string 定位于用坐标轴指定的位置，且对指定的属性进行设置。

【功能介绍】 在当前轴中创建 text 对象。函数 text 是创建 text 图形句柄的低级函数，可用该函数在图形中指定的位置上显示字符串。

【实例 6.10】 绘制正弦曲线并标出零点。

```
>>plot(0:pi/20:2*pi,sin(0:pi/20:2*pi))
>>text(pi,0, 'Zeros Point')
>>grid on
```

得到的图形如图 6.10 所示。

【实例讲解】 text(pi,0, 'Zeros Point')标示了零点的位置及在此位置所显示的文字说明。

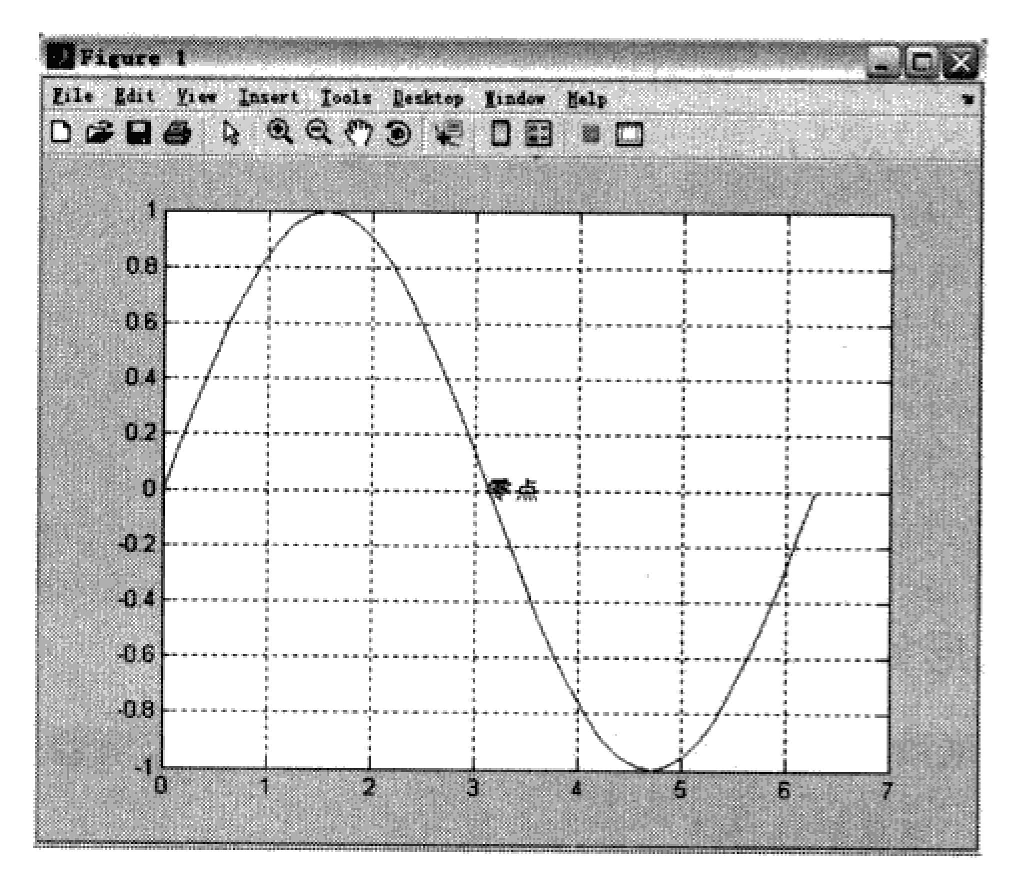


图 6.10　正弦曲线

6.1.7　subplot 函数——分区绘图

【语法说明】 subplot（m,n,p）。

【功能介绍】 该函数将当前图形窗口分成 $m \times n$ 个绘图区，即每行 n 个，共 m 行，区号按行优先编号，且选定第 p 个区为当前活动区。

【实例 6.11】 在一个图形窗口中同时绘制正弦、余弦、正切、余切曲线。

```
x=linspace(0,2*pi,60);
y=sin(x);
z=cos(x);
t=sin(x)./(cos(x)+eps); eps    %为系统内部常数
ct=cos(x)./(sin(x)+eps);
subplot(2,2,1);   %分成 2×2 区域且指定 1 号为活动区
plot(x,y);
title('sin(x)');
axis ([0 2*pi -1 1]);
subplot(2,2,2);
plot(x,z);
```

```
title('cos(x)');
axis ([0 2*pi -1 1]);
subplot(2,2,3);
plot(x,t);
title('tangent(x)');
axis ([0 2*pi -40 40]);
subplot(2,2,4);
plot(x,ct);
title('cotangent(x)');
axis ([0 2*pi -40 40]);
```

得到的图形如图 6.11 所示。

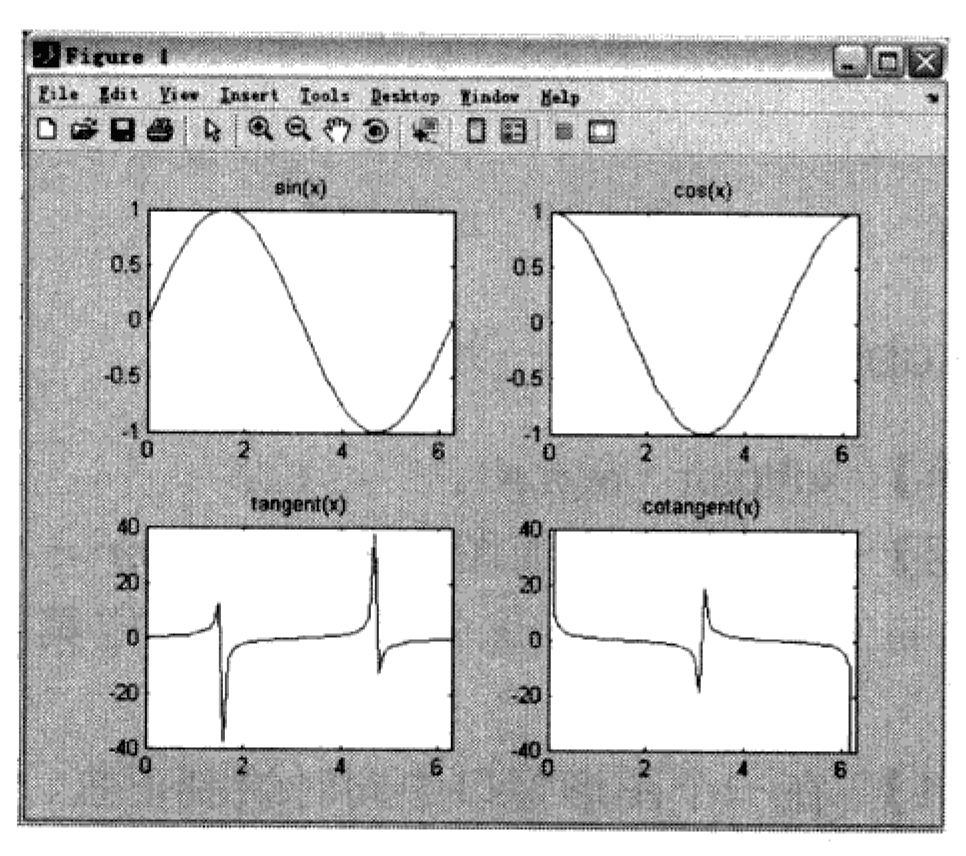


图 6.11　图形的分区显示

【实例讲解】 本例子将一个图形显示界面分成了 4 个部分，每一个部分显示一种曲线图。

6.1.8　grid、box——给坐标加网格和边框

【语法说明】

- grid on/off：给坐标加网格线用 grid 函数来实现。grid on/off 函数控制是画还是不画网格线，不带参数的 grid 函数在两种状态之间进行切换。

- box on/off：给坐标加边框用 box 函数来实现。box on/off 函数控制是加还是不加边框线，不带参数的 box 函数在两种状态之间进行切换。

【功能介绍】 给坐标加网格或边框。

【实例 6.12】 在同一坐标中，绘制 3 个同心圆，并加坐标控制。

```
>>t=0:0.01:2*pi;
>>x=exp(i*t);
>>y=[x;2*x;3*x]';
>>plot(y)
>>grid on;                    %加网格线
>>box on;                     %加坐标边框
>>axis equal                  %坐标轴采用等刻度
```

得到的图形如图 6.12 所示。

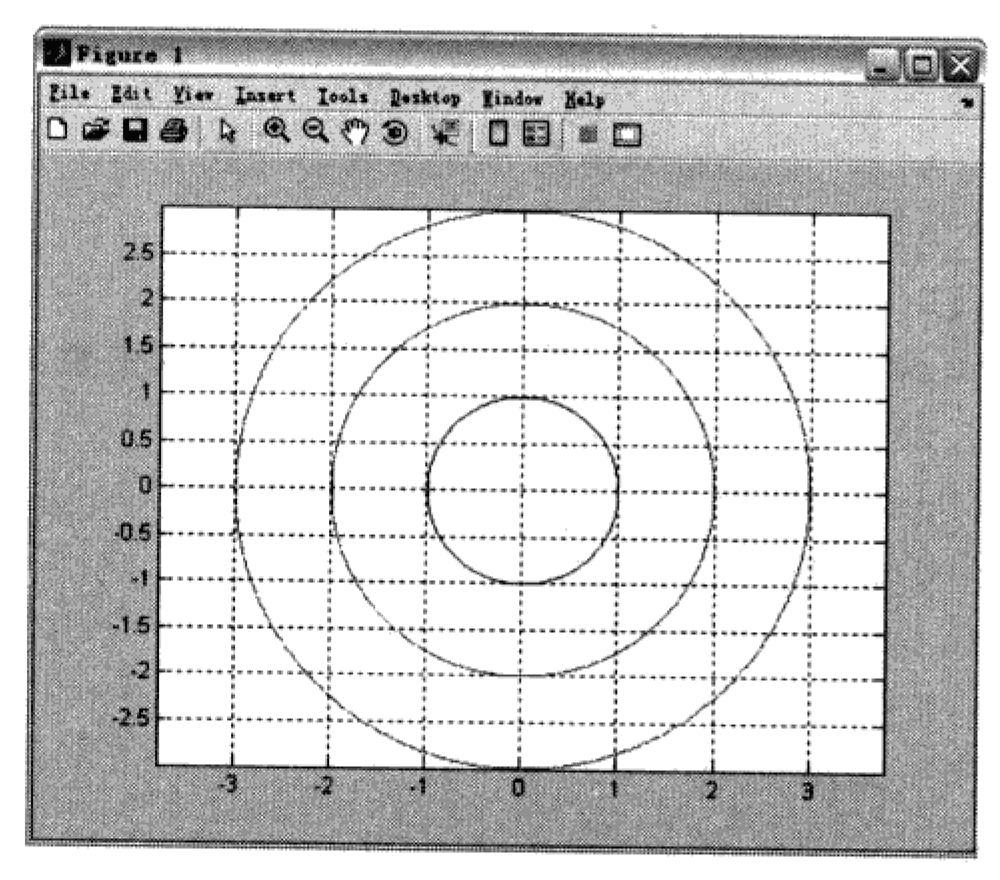


图 6.12 坐标控制

【实例讲解】 使用 grid 函数创建网格，可以让用户更加便利地查看局部图形。

6.1.9 figure 函数——多图形窗口绘制

【语法说明】 H=figure：创建新窗口并返回句柄到变量 H1。

【功能介绍】 需要建立多个图形窗口，绘制并保持每一个窗口的图形，可以使用 figure 函数。每执行一次 figure 函数，就创建一个新的图形窗口，该窗口自动为活动窗口，若需要还可以返回该窗口的识别号码，称该号码为句柄。句柄显示在图形窗口的标题栏中，即图形窗口标题。用户可通过句柄激活或关闭某图形窗口，而 axis、xlabel、title 等许多函数也只对活动窗口有效。

【实例 6.13】 用图形窗口形式重新绘制正弦、余弦、正切和余切曲线。

```
>>x=linspace(0,2*pi,60);
>>y=sin(x);
>>z=cos(x);
>>t=sin(x)./(cos(x)+eps);
>>ct=cos(x)./(sin(x)+eps);
>>H1=figure;     %创建新窗口并返回句柄到变量 H1
>>plot(x,y);      %绘制图形并设置有关属性
>>title('sin(x)');
>>axis ([0 2*pi -1 1]);
>>H2=figure;    %创建第二个窗口并返回句柄到变量 H2
>>plot(x,z);      %绘制图形并设置有关属性
>>title('cos(x)');
>>axis ([0 2*pi -1 1]);
>>H3=figure;      %同上
>>plot(x,t);
>>title('tangent(x)');
>>axis ([0 2*pi -40 40]);
>>H4=figure;    %同上
>>plot(x,ct);
>>title('cotangent(x)');
>>axis ([0 2*pi -40 40]);
```

得到的图形如图 6.13 所示。

【实例讲解】 当用户每次使用 figure 函数的时候，MATLAB 会创建一个新的窗口。

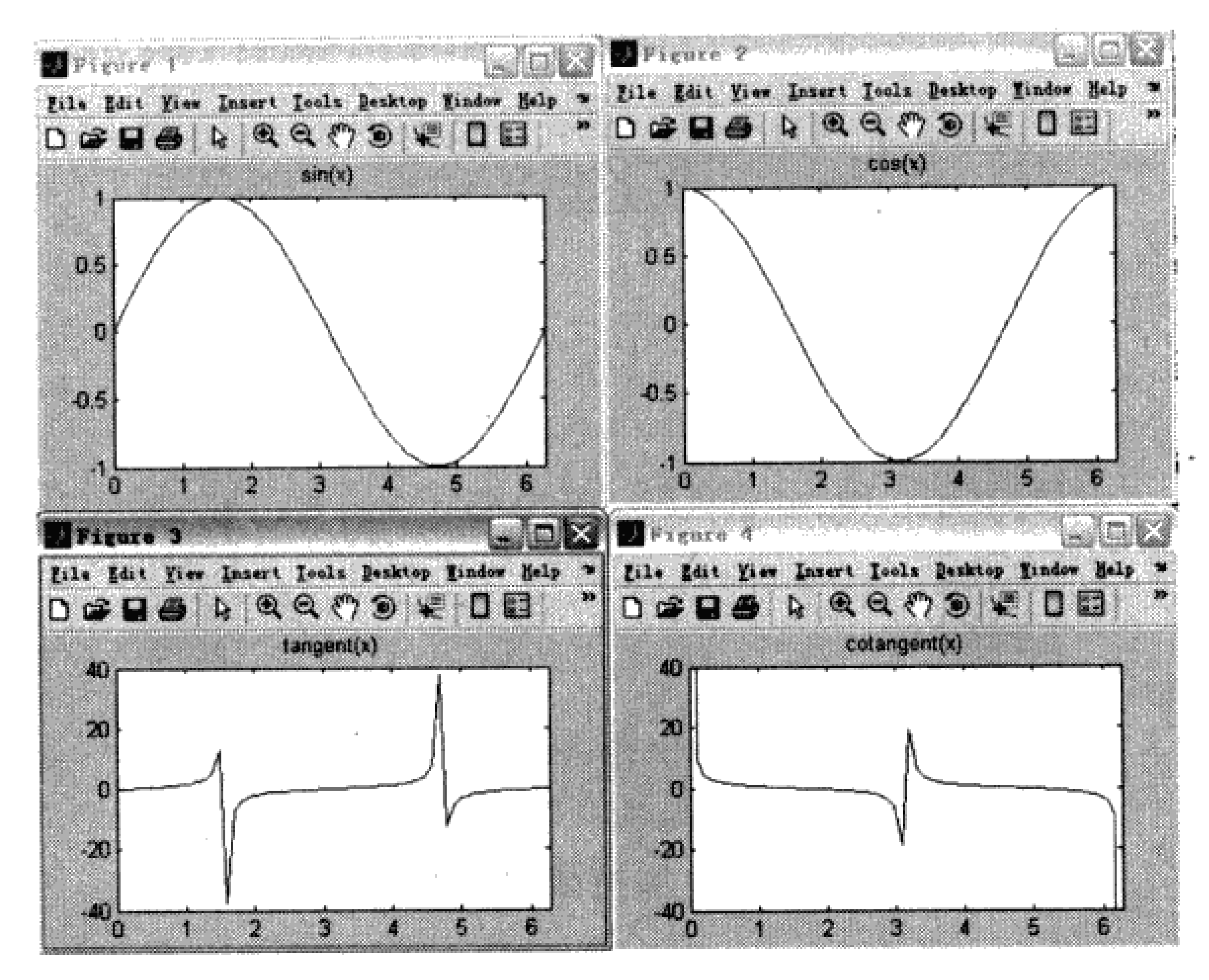


图 6.13　多窗口显示

6.1.10　hold 函数——图形保持

【语法说明】 hold on/off。

【功能介绍】 若在已存在图形窗口中用 plot 函数继续添加新的图形内容，可使用图形保持函数 hold。发出函数 hold on 后，再执行 plot 函数，在保持原有图形或曲线的基础上，添加新绘制的图形。

【实例 6.14】 绘制正弦曲线后再绘制余弦曲线。

```
>>x=linspace(0,2*pi,60);
>>y=sin(x);
>>z=cos(x);
>>plot(x,y,'b');              %绘制正弦曲线
>>hold on;                  %设置图形保持状态
>>plot(x,z,'g');              %保持正弦曲线同时绘制余弦曲线
>>axis ([0 2*pi -1 1]);
>>legend('cos','sin');
>>hold off                  %关闭图形保持
```

得到的图形如图 6.14 所示。

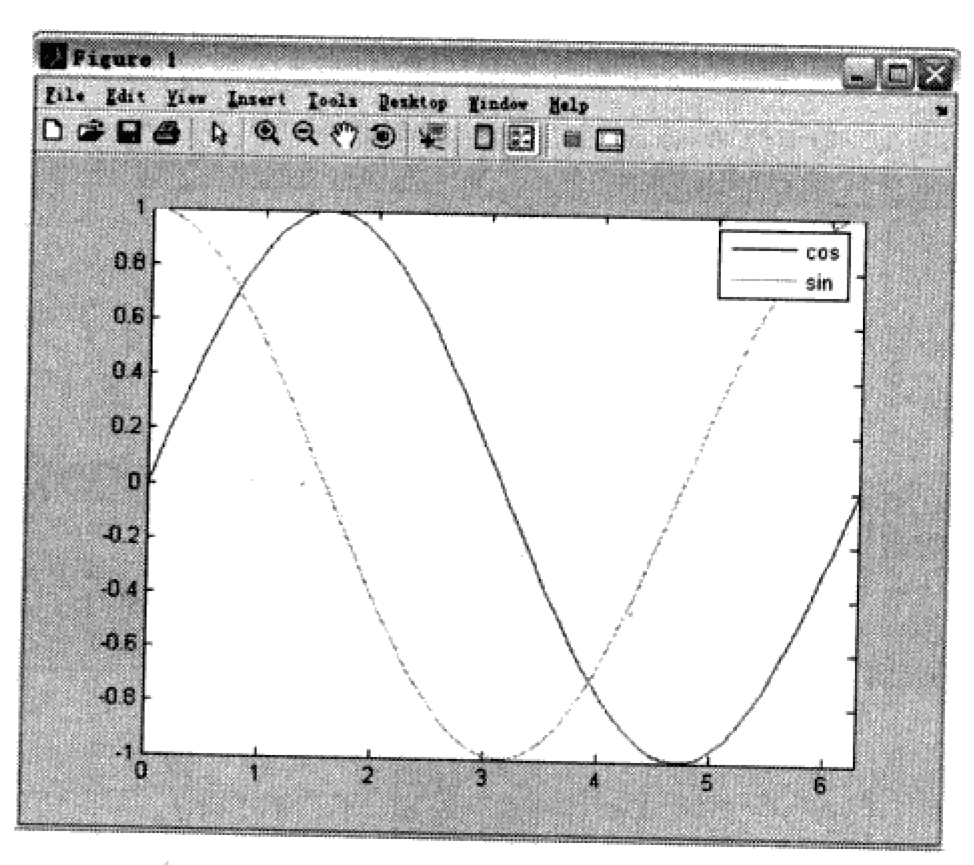


图 6.14 保持函数的使用

【实例讲解】 限于纸面的展示方式，建议用户在 MATLAB 中自行运行上面的函数。

6.1.11 三角图形绘制

【实例 6.15】 利用 M 文件绘制三角函数。

M 文件建立如下所示：

```
function shili02
h0=figure('toolbar','none',...
   'position',[200 150 450 350],...
   'name','实例 02');
x=-pi:0.05:pi;
y=sin(x)+cos(x);
plot(x,y,'-*r','linewidth',1);
grid on
xlabel('自变量 X');
ylabel('函数值 Y');
title('三角函数');
```

得到的图形如图 6.15 所示。

【实例讲解】 得到的图形为在区间 $[-\pi,\pi]$ 上，步长为 0.05 的正余弦曲线之和。

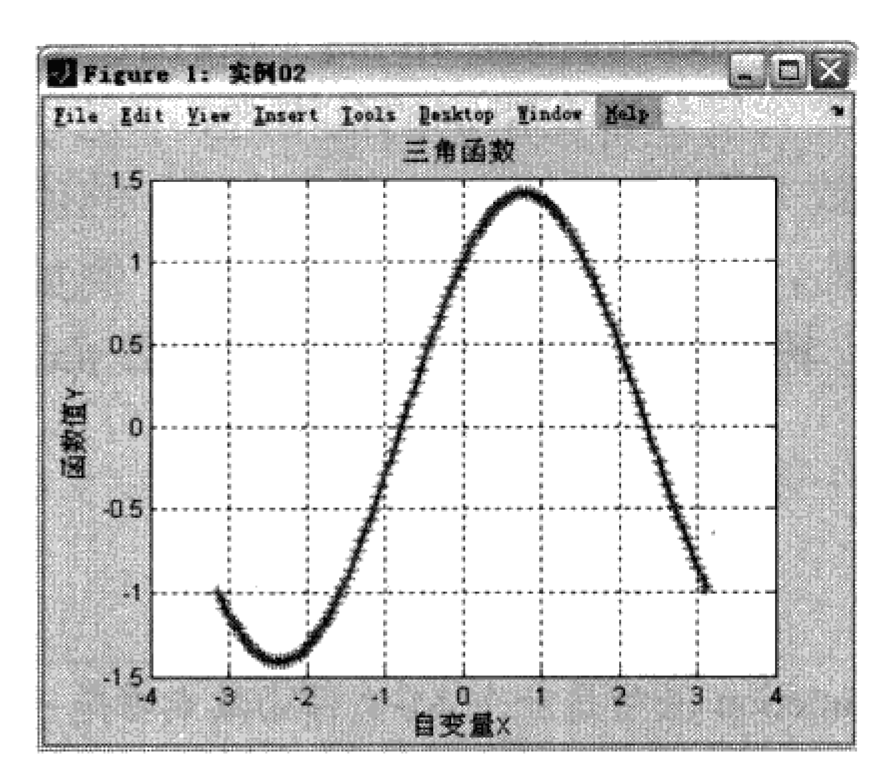


图 6.15 正余弦函数和

6.1.12 fplot——函数 *f*(*x*)曲线

【语法说明】

- fplot('function',limits)：在指定的范围 limits 内画出函数名为 function 的一元函数图形。其中 limits 是一个指定 *x* 轴范围的向量[xmin xmax]或者是 *x* 轴和 *y* 轴的范围的向量[xmin xmax ymin ymax]。
- fplot('function',limits,LineSpec)：用指定的线型 LineSpec 画出函数 function。
- fplot('function',limits,tol)：用相对误差值为 tol 画出函数 function。相对误差的默认值为 2e−3。
- fplot('function',limits,tol,LineSpec)：用指定的相对误差值 tol 和指定的线型 LineSpec 画出函数 function 的图形。
- fplot('function',limits,n)：当 n>=1，则至少画出 n+1 个点（即至少把范围 limits 分成 n 个小区间），最大步长不超过(xmax−xmin)/n。
- fplot('function',limits,…)：允许可选参数 tol，n 和 LineSpec 以任意组合方式输入。
- [X,Y] = fplot('function',limits,⋯)：返回横坐标与纵坐标的值给变量 X 和 Y，此时 fplot 不画出图形，plot(X,Y)可画出图形。

【功能介绍】 在指定的范围 limits 内画出一元函数 $y=f(x)$的图形。其中向量 x 的分量分布在指定的范围内，y 是与 x 同型的向量，对应的分量有函数关系：$y(i)=f(x(i))$。若对应于 x 的值，y 返回多个值，则 y 是一个矩阵，其中每列对应一个 $f(x)$。例如，$f(x)$返回向量 $[f1(x),f2(x),f3(x)]$，输入参量 $x=[x1;x2;x3]$，则函数 $f(x)$返回矩阵

$$\begin{matrix} f1(x1) & f2(x1) & f3(x1) \\ f1(x2) & f2(x2) & f3(x2) \\ f1(x3) & f2(x3) & f3(x3) \end{matrix}$$

注意　函数 function 必须是一个 M-文件函数或者是一个包含变量 x，且能用函数 eval 计算的字符串。

【实例 6.16】 在指定的范围内绘制几种常用的函数图形。

```
>>fplot('tanh',[-2 2])
>>subplot(2,2,1);
>>fplot('humps',[0 1])
>>subplot(2,2,2);
>>fplot('abs(exp(-j*x*(0:9))*ones(10,1))',[0 2*pi])
>>subplot(2,1,2);
>>fplot('[tan(x),sin(x),cos(x)]',2*pi*[-1 1 -1 1])
```

得到的图形如图 6.16 和图 6.17 所示。

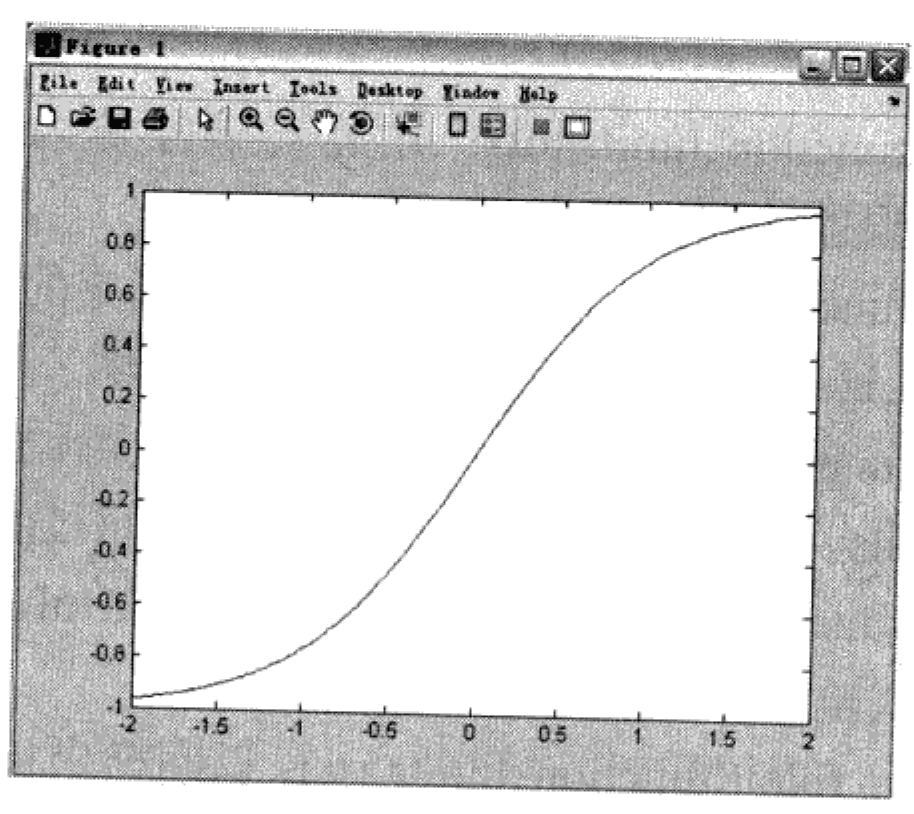


图 6.16　fplot 画图

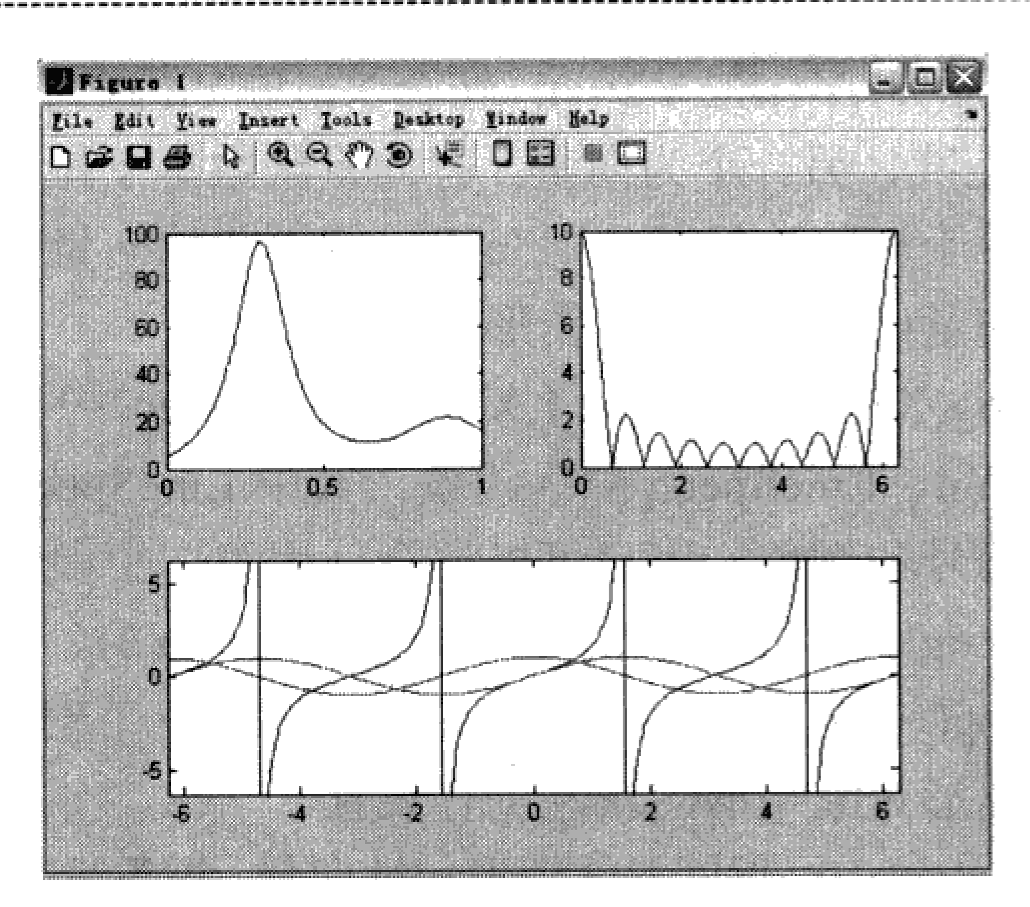


图 6.17 多图形绘制

【实例讲解】 以上几个 fplot 函数的特点是前面位置指明要绘制的函数曲线图形，而后面指明区间范围。

6.2 特殊坐标图形

上一节介绍了普通图形的绘制，这一节将要介绍一些特殊的坐标图形的绘制，包括不同的坐标图形的绘制、特殊的二维图形的绘制以及对图形的修饰修改等操作。在 MATLAB 中提供了一系列函数用来执行这些操作，简单方便，而且绘制图形的效果极好。

6.2.1 loglog 函数——绘制双对数坐标图形

【语法说明】

- loglog(Y)：若 y 为实数向量或矩阵，则结合 y 列向量的下标与 y 的列向量画出。若 y 为复数向量或矩阵，则 loglog(Y)等价于 loglog(real(Y),imag(Y))，在 loglog 的其他使用形式中将忽略 Y 的虚数部分。

- loglog(X1,Y1,X2,Y2…)：结合 Xn 与 Yn 画出图形。若只有 Xn 或 Yn 为矩阵，另一个为向量，行向量维数等于矩阵的列数，列向量的维数等于矩阵的行数，则 loglog 把矩阵按向量的方向分解成向量，再与向量结合分别画出图形。
- loglog(X1,Y1,LineSpec1,X2,Y2,LineSpeec2…)：按顺序取 3 个参数 Xn、Yn、LineSpecn 画出线条，其中 LineSpecn 指定线条的线型、标记符号和颜色。用户可以混合使用第二个参数和第三个参数形式，如：oglog(X1,Y1,X2,Y2,LineSpec2,X3,Y3)。
- loglog(…,'PropertyName',PropertyValue,…)：对所有由 loglog 函数生成的图形对象句柄的属性进行设置。
- h=loglog(…)：返回 line 图形句柄向量，每条线对应一个句柄。

【功能介绍】 绘制对数坐标图形。

【实例 6.17】 绘制对数坐标图。

```
>>x = logspace(-1,2);
>>loglog(x,10*exp(x),'-s')
>>grid on
```

得到的图形如图 6.18 所示。

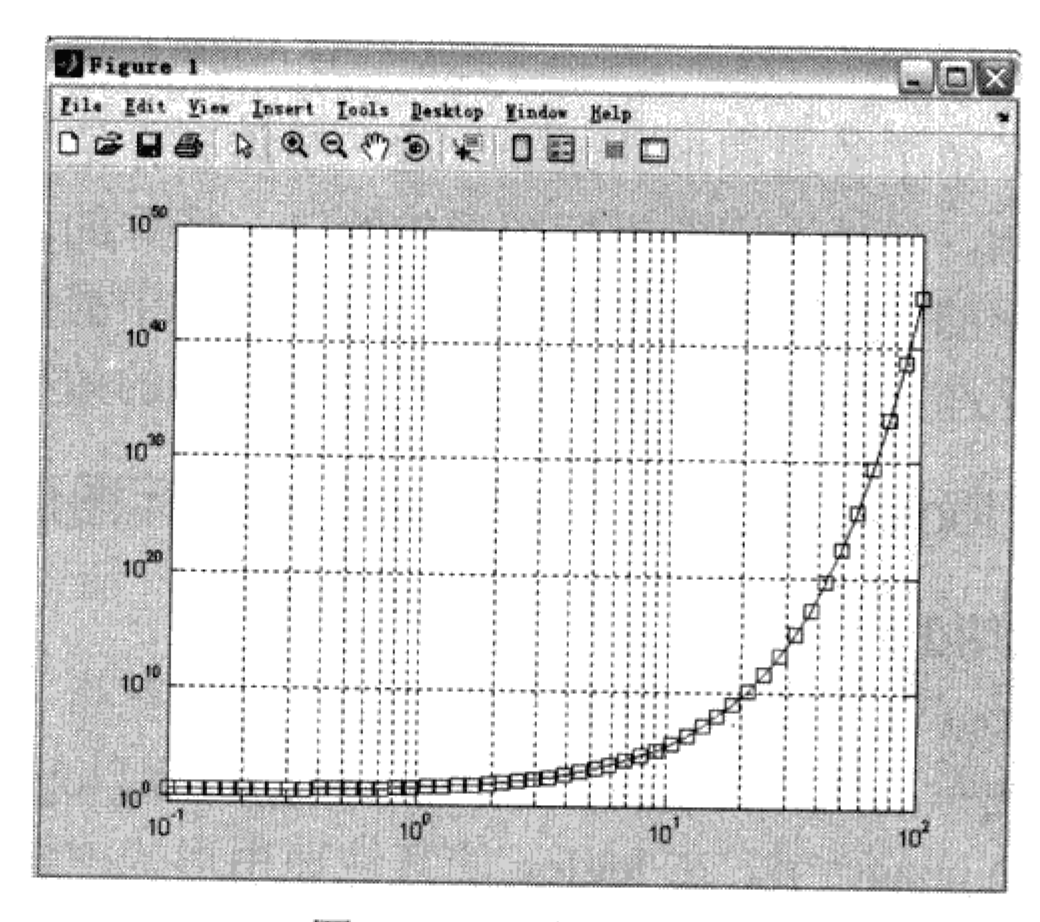


图 6.18　对数坐标图

【实例讲解】 loglog(x,10*exp(x),'-s')中的-s 说明用圆圈的点绘制曲线。

【实例 6.18】 绘制 $y=|1000\sin(4x)|+1$ 的双对数坐标图。

```
x=[0:0.1:2*pi];
y=abs(1000*sin(4*x))+1;
loglog(x,y);
```

得到的图形如图 6.19 所示。

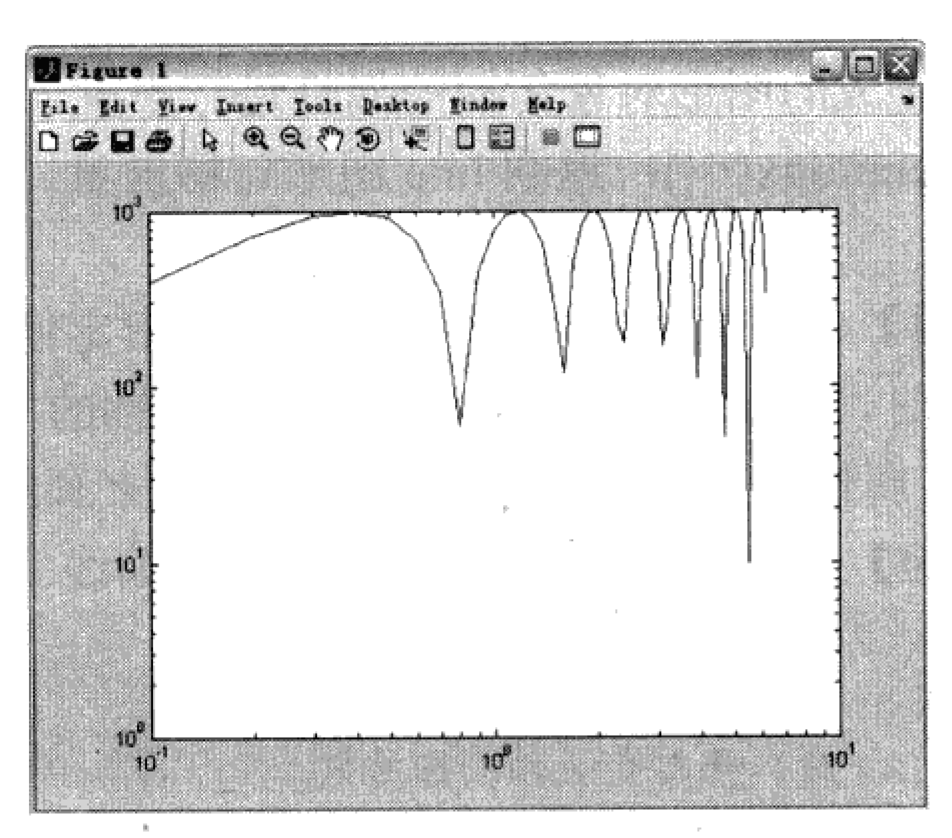


图 6.19　绘制函数的双对数坐标

【实例讲解】 双对数坐标在工程领域中应用广泛。

6.2.2　semilogx 函数——单对数坐标

【语法说明】

- semilogx(Y)：对 x 轴的刻度求常用对数（以 10 为底），而 y 轴为线性刻度。若 Y 为实数向量或矩阵，则结合 Y 列向量的下标与 Y 的列向量画出线条；若 Y 为复数向量或矩阵，则 semilogx(Y) 等价于 semilogx(real(Y),imag(Y))。在 semilogx 的其他使用形式中，Y 的虚数部分将被忽略。
- semilogx(X1,Y1,X2,Y2…)：结合 Xn 和 Yn 画出线条，若其中只有 Xn 或 Yn 为矩阵，另外一个为向量，行向量的维数等于矩阵

的列数，列向量的维数等于矩阵的行数，则按向量的方向分解矩阵，再与向量结合，分别画出线条。

- semilogx(X1,Y1,LineSpec1X2,Y2,LineSpec2,…)：按顺序取 3 参数 Xn、Yn、LineSpecn 画线，参数 LineSpecn 指定使用的线型、标记符号和颜色。用户可以混合使用第 2 个参数和第 3 个参数形式，如：semilogx(X1,Y1,X2,Y2,LineSpec2,X3,Y3)
- semilogx(…,'PropertyName',PropertyValue,…)：对所有由 semilogx 函数生成的图形对象句柄的属性进行设置。
- h = semilogx(…)：返回 line 图形句柄向量，每条线对应一个句柄。

【功能介绍】 绘制 *x* 轴对数图形。若没有指定使用的颜色，当所画线条较多时，semilogx 将自动使用由当前轴的 ColorOrder 和 LineStyleOrder 属性指定的颜色顺序和线型顺序来画线。

【实例 6.19】 以 *x* 轴为对数重新绘制上述曲线。

```
x=[0:0.01:2*pi]
y=abs(1000*sin(4*x))+1
semilogx(x,y); 单对数 x 轴绘图函数
```

得到的曲线如图 6.20 所示。

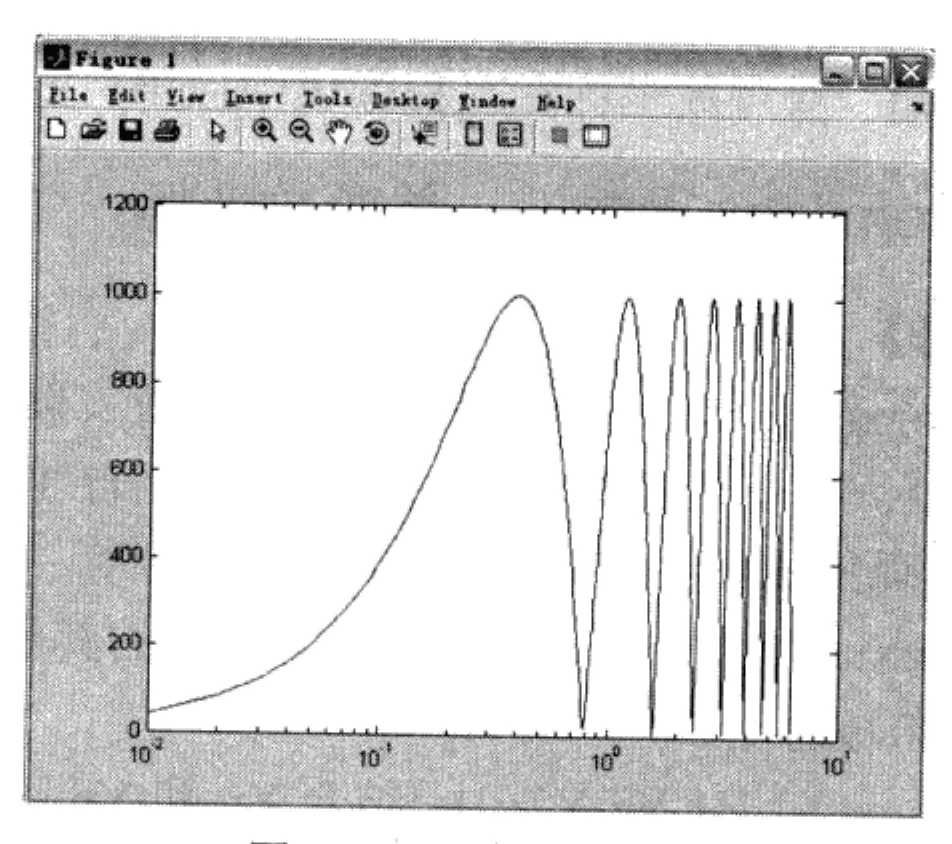


图 6.20　*x* 轴对数曲线

【实例讲解】 semilogx(x,y)表示单对数的以 x 轴为对数绘制。

【实例 6.20】 以 y 轴为对数重新绘制上述曲线。

```
x=[0:0.01:2*pi]
y=abs(1000*sin(4*x))+1
semilogy(x,y); 单对数 y 轴绘图函数
```

以 y 轴为对数绘制的曲线如图 6.21 所示。

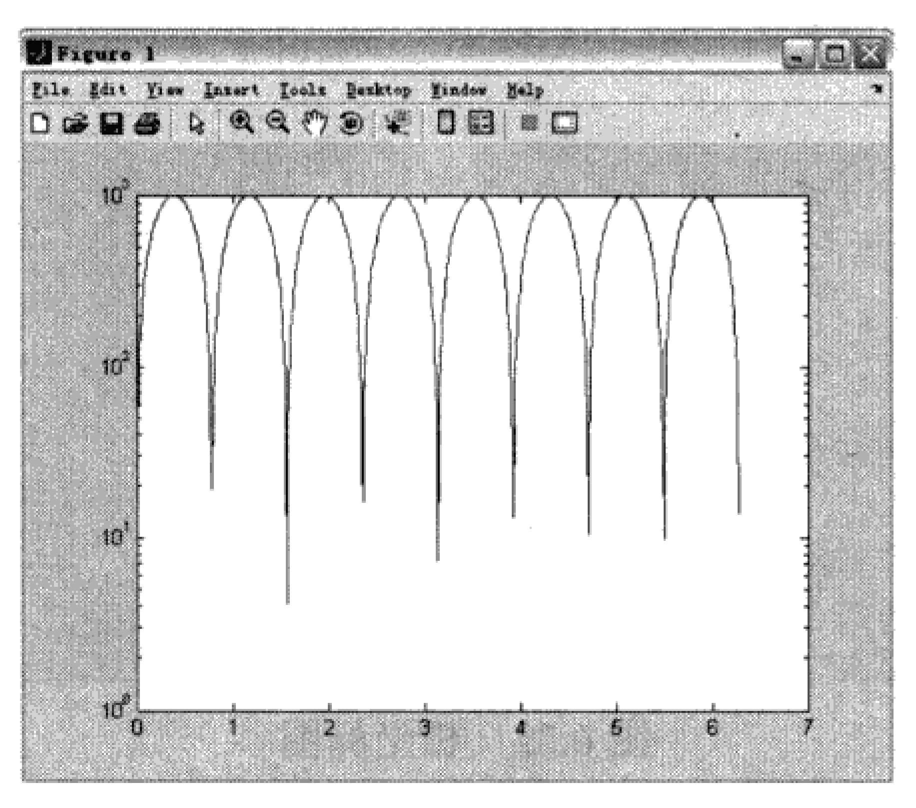


图 6.21　y 轴对数曲线

【实例讲解】 用户可以将本例中的图和上例的图进行对比分析。

6.2.3　polar 函数——绘制极坐标图

【语法说明】

- polar(theta,rho)：用极角 theta 和极径 rho 画出极坐标图形。极角 theta 为从 x 轴到半径的单位为弧度的向量，极径 rho 为各数据点到极点的半径向量。
- polar(theta,rho,LineSpec)：参量 LineSpec 指定极坐标图中线条的线型、标记符号和颜色等。

【功能介绍】 该函数接受极坐标形式的函数 rho=$f(\theta)$，在笛卡儿坐标系平面上画出该函数，且在平面上画出极坐标形式的格栅。

【实例 6.21】 绘制 $\sin(2\times\theta)\times\cos(2\times\theta)$的极坐标图。

```
theta=[0:0.01:2*pi];
rho=sin(2*theta).*cos(2*theta);
polar(theta,rho); %绘制极坐标图函数
title('polar plot');
```

得到的图形如图 6.22 所示。

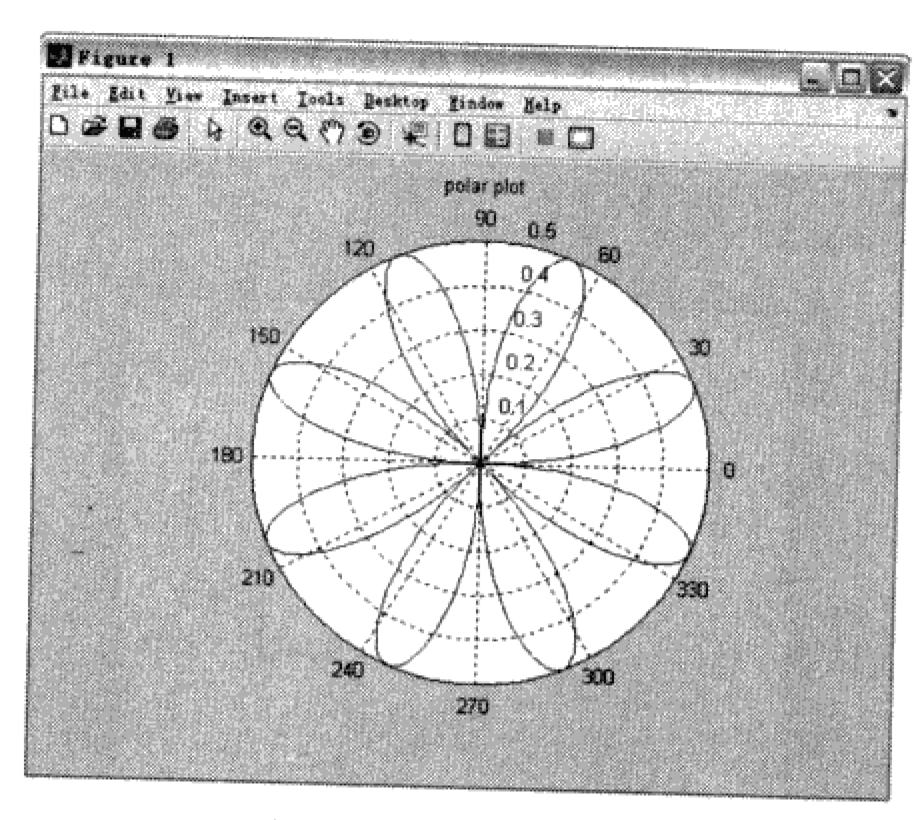


图 6.22 极坐标图

【实例讲解】 polar(theta,rho)中的 theta 指明范围是[0:0.01:2*pi]，rho 指明要绘制的函数为 sin(2*theta).*cos(2*theta)。title('polar plot')添加标题。

6.2.4 bar 函数——二维垂直条形图

【语法说明】

- barh(Y)：若 y 为向量，则分别显示每个分量的高度，纵坐标为 1 到 length(y)；若 y 为矩阵，则 bar 把 y 分解成行向量，再分别画出，纵坐标为 1 到 size(y,1)，即矩阵的行数。
- bar(x,Y)：在指定的纵坐标 x 上以水平方向画出 y，其中 x 为严格单增的向量。若 y 为矩阵，则 bar 把矩阵分解成几个行向量，在指定的纵坐标处分别画出。
- bar(…,width)：设置条形的相对宽度和控制在一组内条形的

间距。默认值为 0.8，所以，如果用户没有指定 x，则同一组内的条形有很小的间距；若设置 width 为 1，则同一组内的条形相互接触。

- bar(…,'style')：指定条形的排列类型。类型有“group”和“stack”，其中“group”为默认的显示模式。“group”：若 y 为 $n\times m$ 阶的矩阵，则 bar 显示 n 组，每组有 m 个水平条形的条形图。“stack”：对矩阵 y 的每一个行向量显示在一个条形中，条形的高度为该行向量中的分量和。其中同一条形中的每个分量用不同的颜色显示出来，从而可以显示每个分量在向量中的分布。
- bar(…,LineSpec)：用指定的颜色 LineSpec 显示所有的条形。
- [xb,yb] = bar(…)：返回用户可用函数 plot 或函数 patch 画出条形图的参量 xb、yb。这对用户控制一个图形的显示是有用的，例如要在一个 plot 语句中加入装饰性的条形图等。

【功能介绍】 二维垂直条形图。用垂直条形显示向量或矩阵中的值。

【实例 6.22】 用二维垂直条形图绘制函数曲线图。

```
x = -2.9:0.2:2.9;
bar(x,exp(x.*sin(x)))
colormap gray
```

得到的图形如图 6.23 所示。

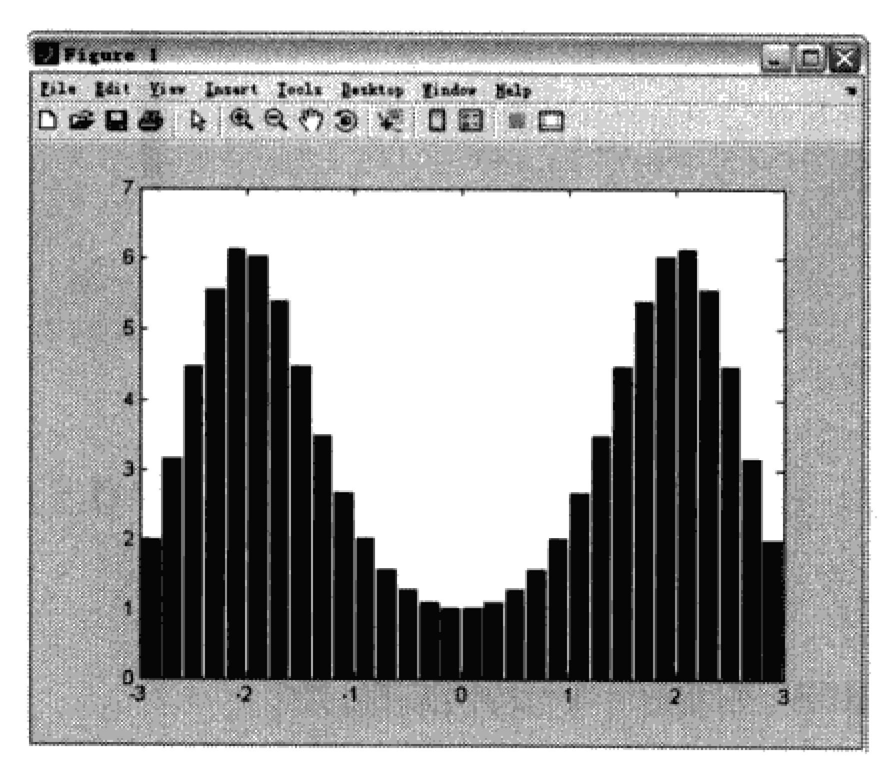


图 6.23 条形图

【实例讲解】 x 指定区间范围[-2.9，2.9]步长为 0.2。

6.2.5 barh 函数——二维水平条形图

【语法说明】 h = barh(…)：返回一个 patch 图形对象句柄的向量。每一条形对应一个句柄。

【功能介绍】 二维水平条形图。用水平条形显示向量或矩阵中的值。

【实例 6.23】 绘制水平条形图。

```
>>X = 1:.5:5;
>>Y = exp(X).*sin(X);
>>barh(Y,'stack')
```

得到的图形如图 6.24 所示。

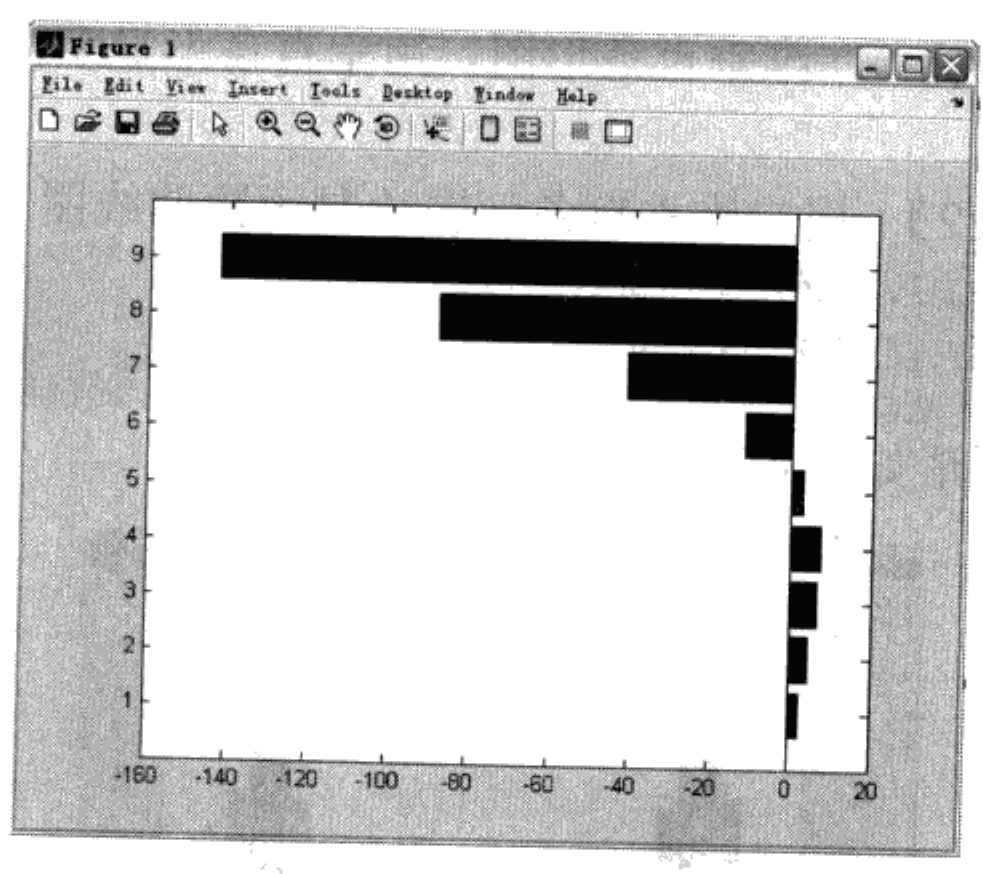


图 6.24 水平条形图

【实例讲解】 x 指定区间范围[1，5]步长为 0.5，绘制的函数与上例相同。

6.2.6 stairs 函数——阶梯图形

【语法说明】 stairs(x,y)：x,y 分别为横坐标向量和纵坐标向量。

【功能介绍】 绘制阶梯图形。

【实例 6.24】 绘制阶梯图形。

```
x=[-2.5:0.25:2.5];
y=exp(-x.*x);
stairs(x,y);  %绘制阶梯图形函数
title('stairs plot');
```

得到的图形如图 6.25 所示。

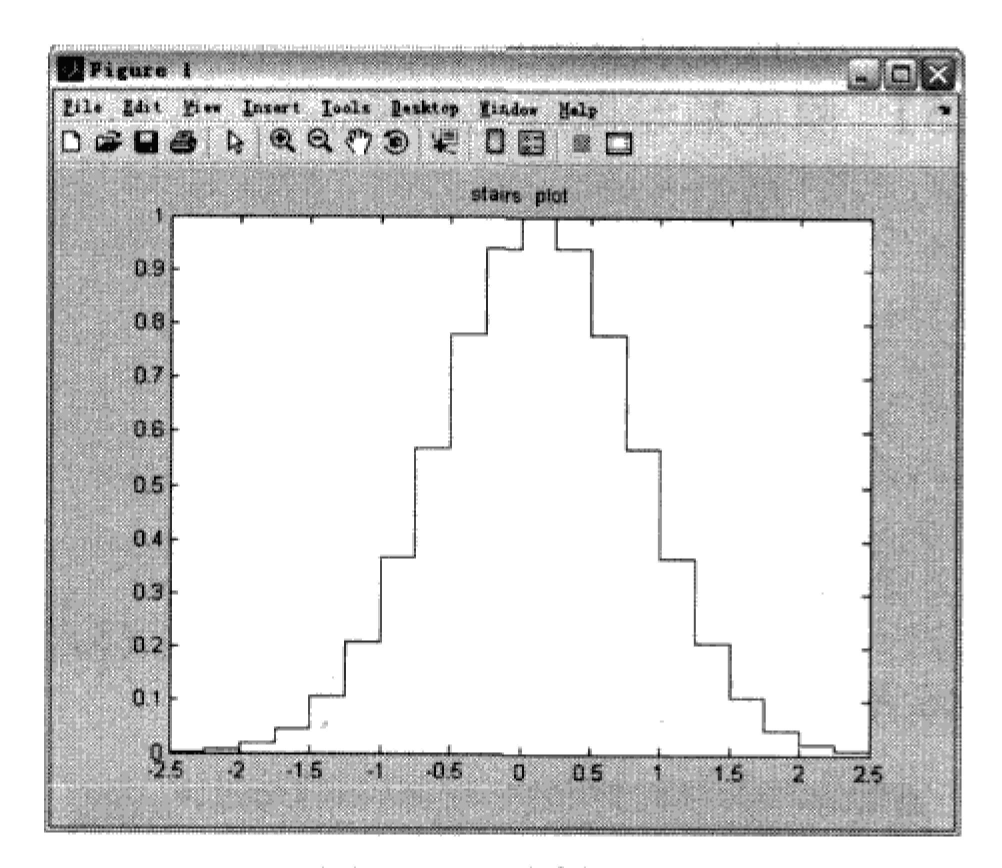


图 6.25 阶梯形图

【实例讲解】 从阶梯图中可以看出每个局部的变化。

6.2.7 ezplot 函数——隐函数图形绘制

【语法说明】

- 对于函数 $f=f(x)$，ezplot 函数的调用格式为：

ezplot(f)：在默认区间 $-2\pi<x<2\pi$ 绘制 $f=f(x)$ 的图形。

ezplot(f, [a,b])：在区间 $a<x<b$ 绘制 $f=f(x)$ 的图形。

- 对于隐函数 $f=f(x,y)$，ezplot 函数的调用格式为：

ezplot(f)：在默认区间 $-2\pi<x<2\pi$ 和 $-2\pi<y<2\pi$ 绘制 $f(x,y)=0$ 的图形。

ezplot(f, [xmin,xmax,ymin,ymax])：在区间 $xmin<x<xmax$ 和

$ymin<y<ymax$ 绘制 $f(x,y) = 0$ 的图形。

ezplot(f, [a,b])：在区间 $a<x<b$ 和 $a<y<b$ 绘制 $f(x,y)=0$ 的图形。

- 对于参数方程 $x = x(t)$和 $y = y(t)$，ezplot 函数的调用格式为：

ezplot(x,y)：在默认区间 $0<t<2\pi$ 绘制 $x=x(t)$和 $y=y(t)$的图形。

ezplot(x,y, [tmin,tmax])：在区间 $tmin<t<tmax$ 绘制 $x=x(t)$和 $y=y(t)$的图形。

【功能介绍】 绘制隐函数图形。

【实例 6.25】 隐函数绘图应用举例。

```
subplot(2,2,1);
ezplot('x^2+y^2-9');
axis equal
subplot(2,2,2);
ezplot('x^3+y^3-5*x*y+1/5')
subplot(2,2,3);
ezplot('cos(tan(pi*x))',[ 0,1])
subplot(2,2,4);
ezplot('8*cos(t)','4*sqrt(2)*sin(t)',[0,2*pi])
```

绘制的 4 幅图形如图 6.26 所示。

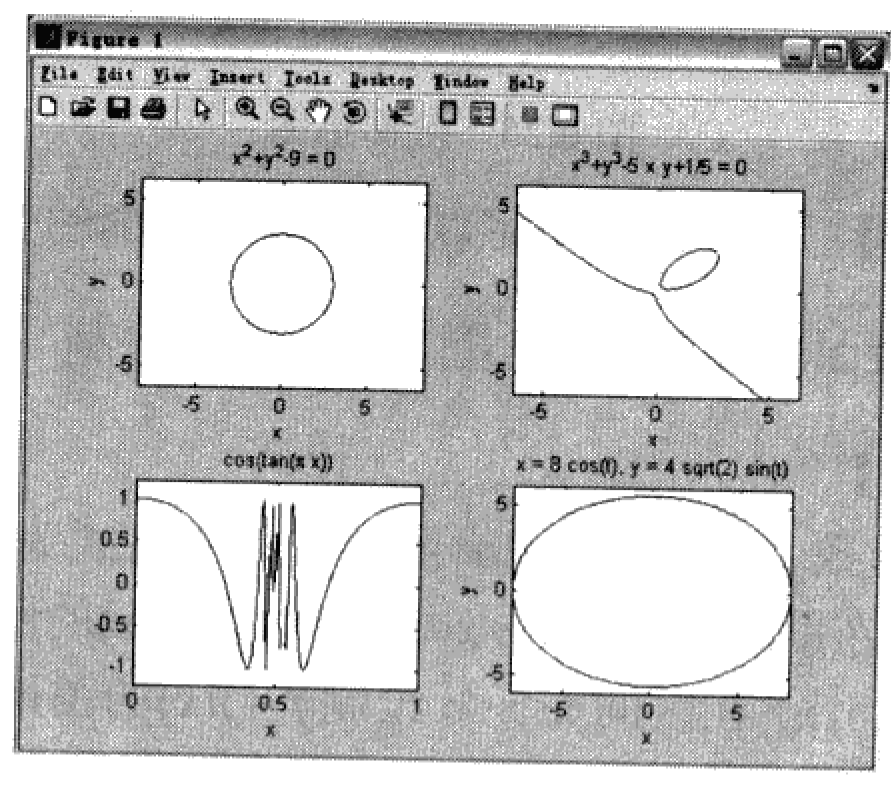


图 6.26 隐函数绘制

【实例讲解】 subplot(2,2,1)用来指定在 4 幅图中，当前这幅图所在的位置，ezplot('x^2+y^2-9')函数中直接指明隐函数。

6.2.8　fill 函数——填充图形

【语法说明】

- fill(X,Y,C)：用 X 和 Y 中的数据生成多边形，用 C 指定的颜色填充它。其中 C 为色图向量或矩阵。若 C 是行向量，则要求 C 的维数等于 X 和 Y 的列数；若 C 为列向量，则要求 C 的维数等于 X 和 Y 的行数。
- fill(X,Y,ColorSpec)：用 ColorSpec 指定的颜色填充由 X 和 Y 定义的多边形。
- fill(X1,Y1,C1,X2,Y2,C2,…)：指定多个要填充的二维区域。
- fill(…,'PropertyName',PropertyValue)：允许用户对一个 patch 图形对象的某个属性设定属性值。
- h = fill(…)：返回 patch 图形对象句柄的向量，每一个 patch 对象对应一个句柄。

【功能介绍】 用颜色填充二维多边形。

【实例 6.26】 用黑颜色填充图形。

```
>>t = (1/16:1/8:1)'*2*pi;
>>x = exp(t).*sin(t);
>>y = t.*cos(t);
>>fill(x,y,'k')
```

得到的图形如图 6.27 所示。

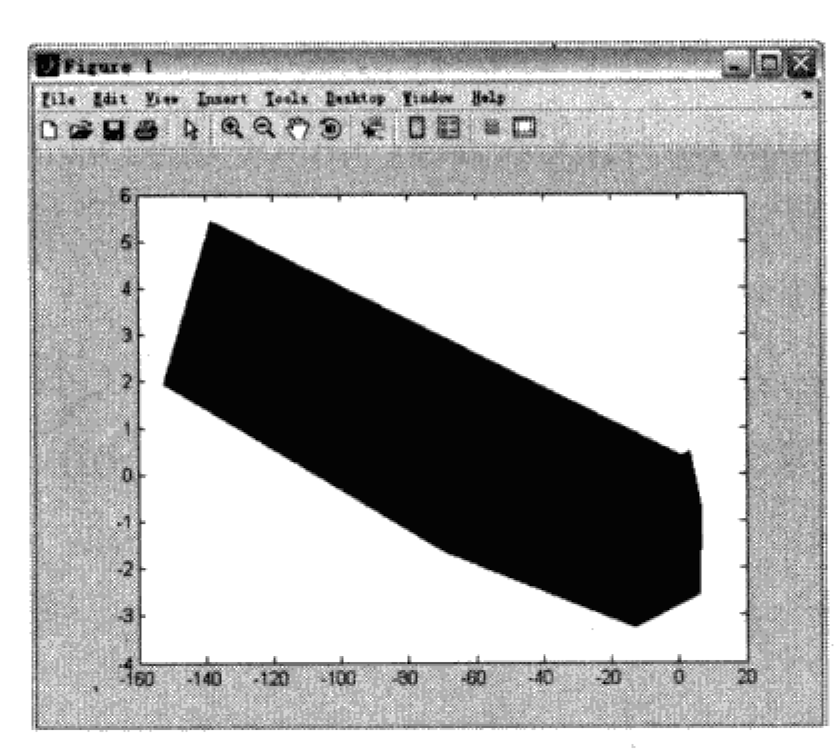


图 6.27　填充图形

【实例讲解】 用户可以使用 fill 函数来填充各种二维图形。

6.2.9 zoom 函数——对图形缩放

【语法说明】

- zoom on：打开交互式的放大功能。当一个图形处于交互式的放大状态时，有两种方法来放大图形。对于一键鼠标、二键或三键鼠标，单击坐标轴内的任意一点，可使图形放大一倍，这一操作可进行多次，直到 MATLAB 的最大显示为止；对于二键或三键的鼠标，在坐标轴内单击右键，可使图形缩小一倍，这一操作可进行多次，直到还原图形为止。对于一键鼠标，要想缩小图形，需要按住键盘上的 Shift 键，再单击鼠标键。用鼠标拖出要放大的部分，系统将放大选定的区域。
- zoom off：关闭交互式放大功能。
- zoom out：将系统转回非放大状态，并将图形恢复原状。
- zoom reset：系统将记住当前图形的放大状态，作为放大状态的设置值。以后使用 zoom out 或者是双击鼠标时，交互式放大状态打开，且图形并不是返回到原状，而是返回 reset 时的放大状态。
- zoom：用于切换放大的状态：on 和 off。
- zoom xon：只对 x 轴进行放大。
- zoom yon：只对 y 轴进行放大。
- zoom(factor)：用放大系数 factor 进行放大或缩小，而不影响交互式放大的状态。若 factor>1，系统将图形放大 factor 倍，若 0<factor≤1，系统将图形放大 1/factor 倍。
- zoom(fig, option)：指定对窗口 fig 中（不一定为当前窗口）的二维图形进行放大，其中参数 option 为 on、off、xon、yon、reset、factor 等。

【功能介绍】 对二维图形进行放大或缩小，放大或缩小会改变坐标轴范围。

【实例 6.27】 绘制箭形图，并对其进行放大操作。

```
>> Z = magic(20).*randn(20);
>> compass(Z)
>> zoom on
```

原图形与放大后的图形如图 6.28 和图 6.29 所示。

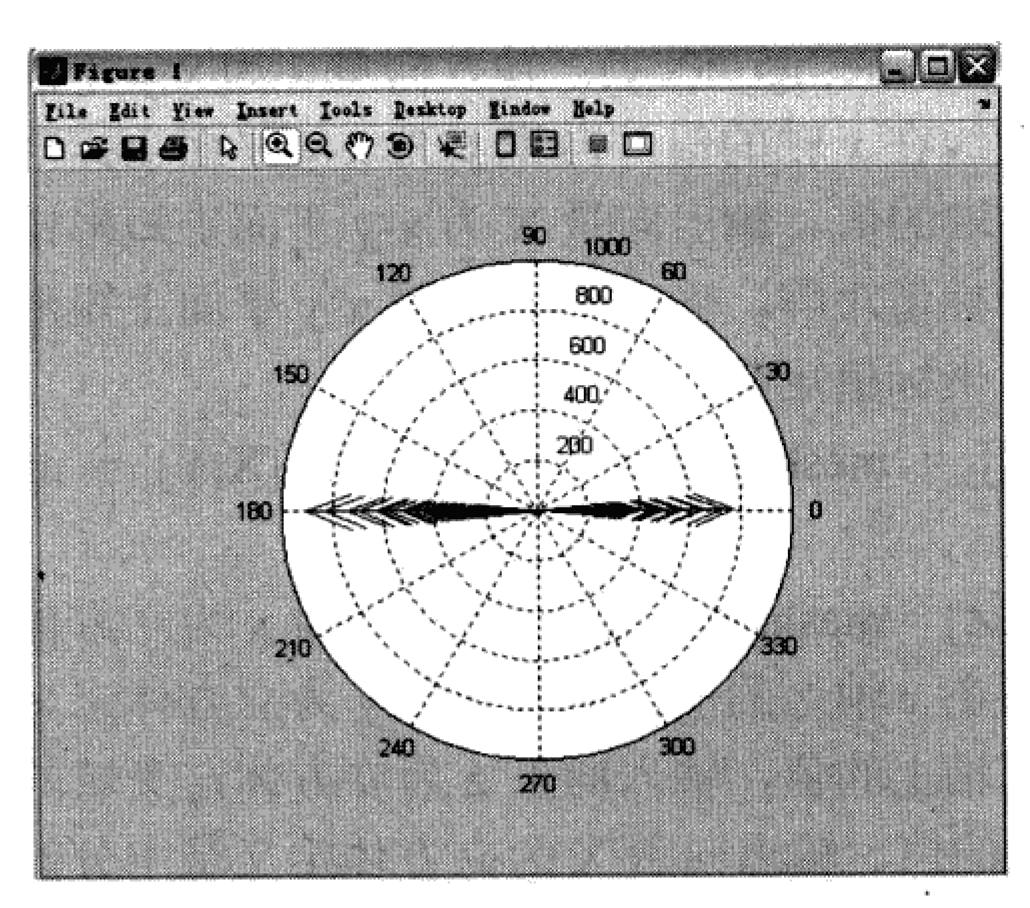


图 6.28　原来图形

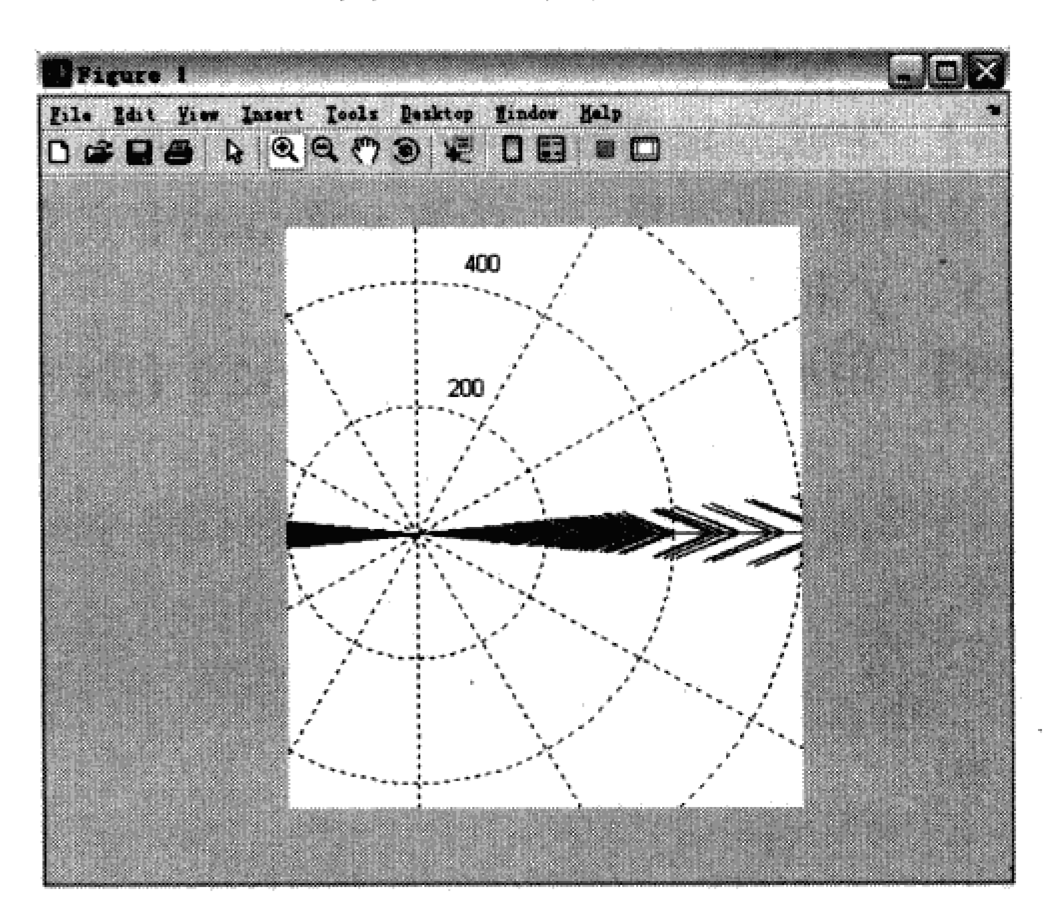


图 6.29　其对应的放大图形

【实例讲解】 上例演示的是对箭形图进行放大的操作。

6.2.10 meshgrid 函数——生成数据点矩阵

【语法说明】

- [X,Y] = meshgrid(x,y)：输入向量 x 为 x-y 平面上矩形定义域的矩形分割线在 *x* 轴的值，向量 y 为 x-y 平面上矩形定义域的矩形分割线在 *y* 轴的值。输出向量 X 为 x-y 平面上矩形定义域的矩形分割点的横坐标值矩阵，输出向量 Y 为 x-y 平面上矩形定义域的矩形分割点的纵坐标值矩阵。
- [X,Y] = meshgrid(x)：等价于形式[X,Y] = meshgrid(x) = meshgrid(x,x)。
- [X,Y,Z] = meshgrid(x,y,z)：输入向量 x 为立方体定义域的立方体分割平面在 *x* 轴上的值，输入向量 y 为立方体定义域的立方体分割平面在 *y* 轴上的值，输入向量 z 为立方体定义域的立方体分割平面在 *z* 轴上的值。输出向量 X 为立方体定义域中分割点的 *x* 轴坐标值，Y 为立方体定义域中分割点的 *y* 轴坐标值，Z 为立方体定义域中分割点的 *z* 轴坐标值。

【功能介绍】 生成二元函数 $z=f(x,y)$中 *x-y* 平面上的矩形定义域中数据点矩阵 X 和 Y，或者是三元函数 $u=f(x,y,z)$中立方体定义域中的数据点矩阵 X、Y 和 Z。

【实例 6.28】 用向量 *x* = [0.7 1.1]; *y* = [-2 3 1]; *z* = [2 5 3]生成数据点矩阵。

```
>>x = [0.7 1.1 ]; y = [-2 3 1]; z = [2 5 3];  %分量不一定从小到大
>>[X_2d,Y_2d] = meshgrid(x,y)
>>[X_3d,Y_3d,Z_3d] = meshgrid(x,y,z)
```

计算结果为：

```
X_2d =
    0.7000    1.1000
    0.7000    1.1000
    0.7000    1.1000
Y_2d =
```

```
    -2    -2
     3     3
     1     1
X_3d(:,:,1) =
    0.7000    1.1000
    0.7000    1.1000
    0.7000    1.1000
X_3d(:,:,2) =
    0.7000    1.1000
    0.7000    1.1000
    0.7000    1.1000
X_3d(:,:,3) =
    0.7000    1.1000
    0.7000    1.1000
    0.7000    1.1000
Y_3d(:,:,1) =
    -2    -2
     3     3
     1     1
Y_3d(:,:,2) =
    -2    -2
     3     3
     1     1
Y_3d(:,:,3) =
    -2    -2
     3     3
     1     1
Z_3d(:,:,1) =
     2     2
     2     2
     2     2
Z_3d(:,:,2) =
     5     5
     5     5
     5     5
Z_3d(:,:,3) =
     3     3
     3     3
     3     3
```

【实例讲解】 meshgrid 函数是 MATLAB 中绘制二维和三维图

形的基础函数，用来生成绘制图形的数据点矩阵。

6.2.11 compass 函数——从原点画箭头图

【语法说明】

- compass(X,Y)：参量 x 与 y 为同型的 *n* 维向量，则函数显示 *n* 个箭头，箭头的起点为原点，箭头的位置为[X(i),Y(i)]。
- compass(Z)：参量 z 为 *n* 维复数向量，则函数显示 *n* 个箭头，箭头起点为原点，箭头的位置为[real(Z),imag(Z)]。
- compass(…,LineSpec)：用参量 LineSpec 指定箭头图的线型、标记符号、颜色等属性。
- h = compass(…) 返回 line 对象的句柄给 h。

【功能介绍】 从原点画箭头图。

【实例 6.29】 从原点画箭头图。

```
Z = magic(20).*randn(20);
compass(Z)
```

得到的结果图如图 6.30 所示。

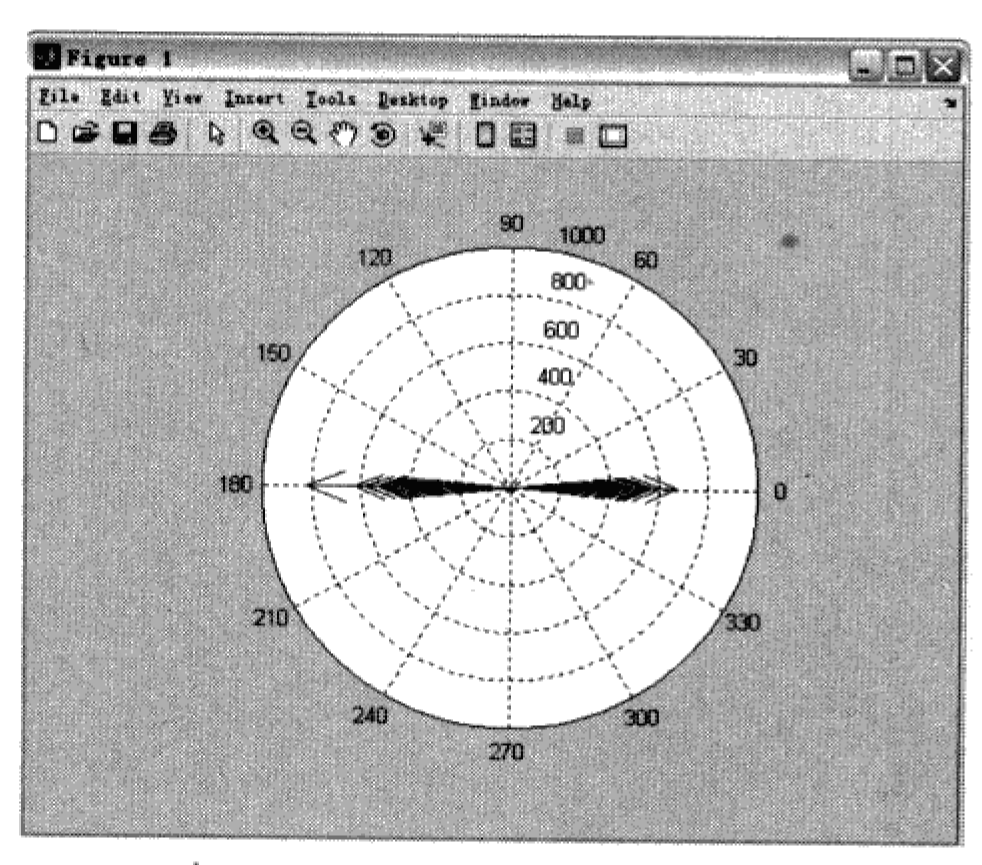


图 6.30 箭头图绘制

【实例讲解】 在上面的函数中，先用 magic(20).*randn(20)生成

随机阵列，而后画箭头图。

6.2.12　comet 函数——绘制二维彗星图

【语法说明】

- comet(y)：彗星图动画显示向量 y 确定的路线。
- comet(x,y)：彗星图动画显示向量 x 与 y 确定的路线。
- comet(x,y,p)：指定彗星体的长度 p*length(y)，默认的 p 值为 0.1。

【功能介绍】 绘制二维彗星图。彗星图为彗星头（一个小圆圈）沿着数据点前进的动画，彗星体为跟在彗星头后面的痕迹，轨道为沿着整个函数的实线。

【实例 6.30】 绘制彗星图。

```
>>t = 0:.01:2*pi;
>>x = exp(sin(2*t)).*(cos(t).^2/3);
>>y = t.*(sin(t).^2);
>>comet(x,y);
```

图形结果如图 6.31 所示。

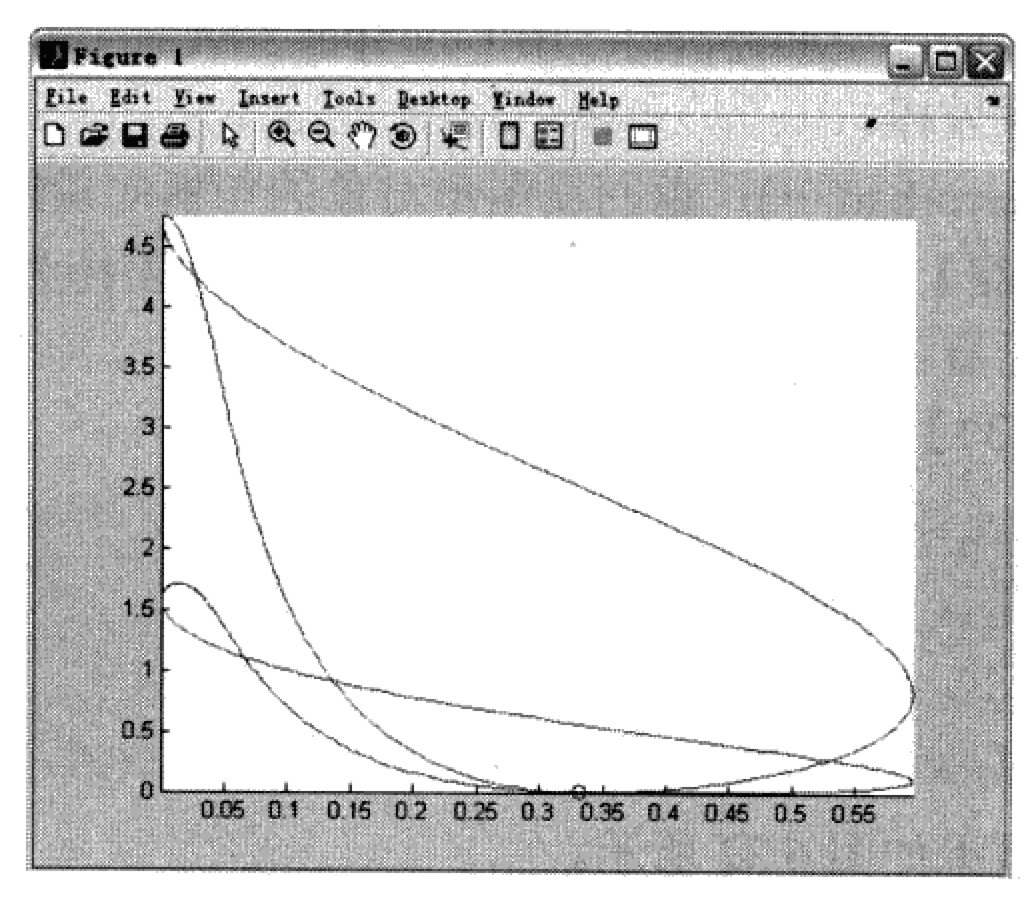


图 6.31　动画显示彗星图绘制

【实例讲解】 x、y 分别指定横坐标和纵坐标。

6.2.13 errorbar 函数——绘制误差图

【语法说明】

- errorbar(Y,E)：画出向量 y，同时显示在向量 y 的每一元素之上的误差棒。误差棒为 E(i)在曲线 y 上面与下面的距离，误差棒的长度为 2*E(i)。
- errorbar(X,Y,E)：X、Y、E 必须为同型参量。若同为向量，则画出带长度为 2*E(i)、对称误差棒于曲线点(X(i),Y(i))之处；若同为矩阵，则画出带长度为 E(i，j)、对称误差棒于曲面点(X(i,j),Y(i,j))之处。
- errorbar(X,Y,L,U)：X、Y、L、U 必须为同型参量。若同为向量，则在点(X(i),Y(i))处画出向下长为 L(i)，向上长为 U(i)的误差棒；若同为矩阵，则在点(X(i,j),Y(i.j))处画出向下长为 L(i,j)，向上长为 U(i,j)的误差棒。
- errorbar(…,LineSpec)：用 LineSpec 指定的线型、标记符、颜色等画出误差棒。
- h = errorbar(…)：返回线图形对象的句柄向量给 h。

【功能介绍】 沿着一曲线画误差棒形图。误差棒为数据的置信水平或者为沿着曲线的偏差。

【实例 6.31】 绘制误差棒图。

```
>>x = 0:pi/10:pi;
>>y= exp(x).*sin(x);
>>e= std(y)*ones(size(x));
>>errorbar(x,y,e)
```

得到的结果如图 6.32 所示。

【实例讲解】 e 表示的是误差，沿着曲线 y 画出对称的误差。

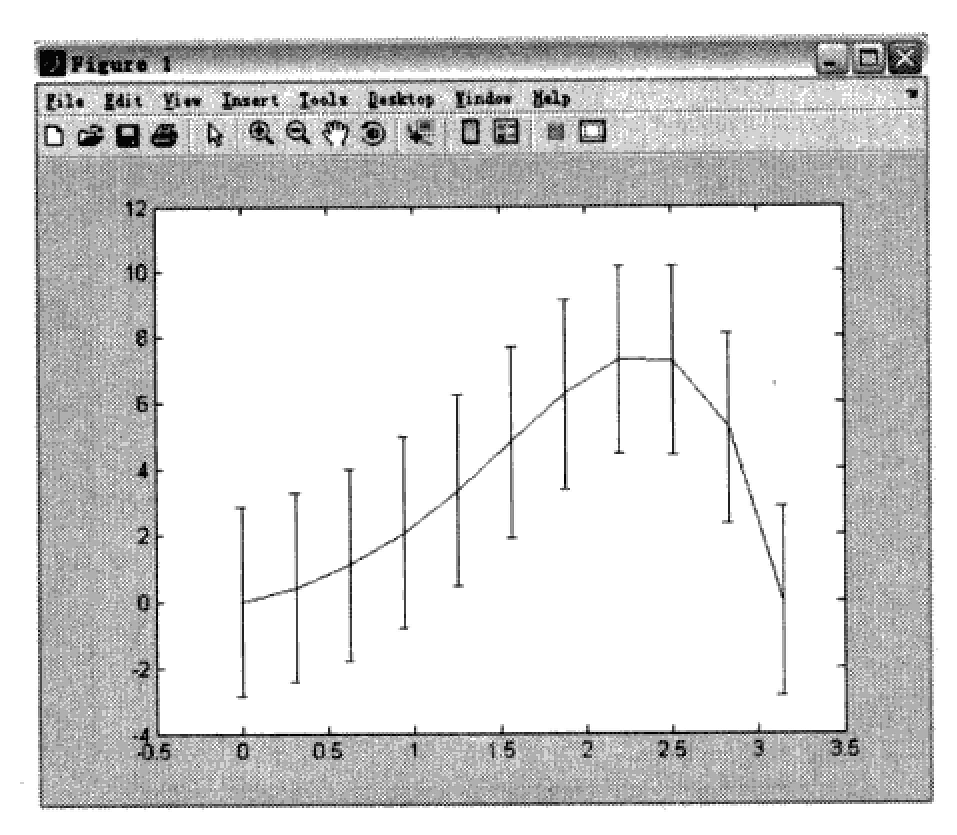


图 6.32 误差棒形图

6.2.14 feather 函数——画速度向量图

【语法说明】

- feather(U,V)：显示由参量向量 u 与 v 确定的向量，其中 u 包含作为相对坐标系中的 x 成分，v 包含作为相对坐标系中的 y 成分。
- feather(Z)：显示复数参量向量 z 确定的向量，等价于 feather(real(Z),imag(Z))。
- feather(…,LineSpec)：用参量 LineSpec 指定的线型、标记符号、颜色等属性画出羽毛图。

【功能介绍】 画出速度向量图。用户要表示各个向量的、相对于原点的向量分量。

【实例 6.32】 绘制速度向量图。

```
>>th = (-90:10:90)*pi/180;
>>r = 4*ones(size(th));
>>[u,v] = pol2cart(th,r);
>>feather(u,v);
```

得到的速度向量图如图 6.33 所示。

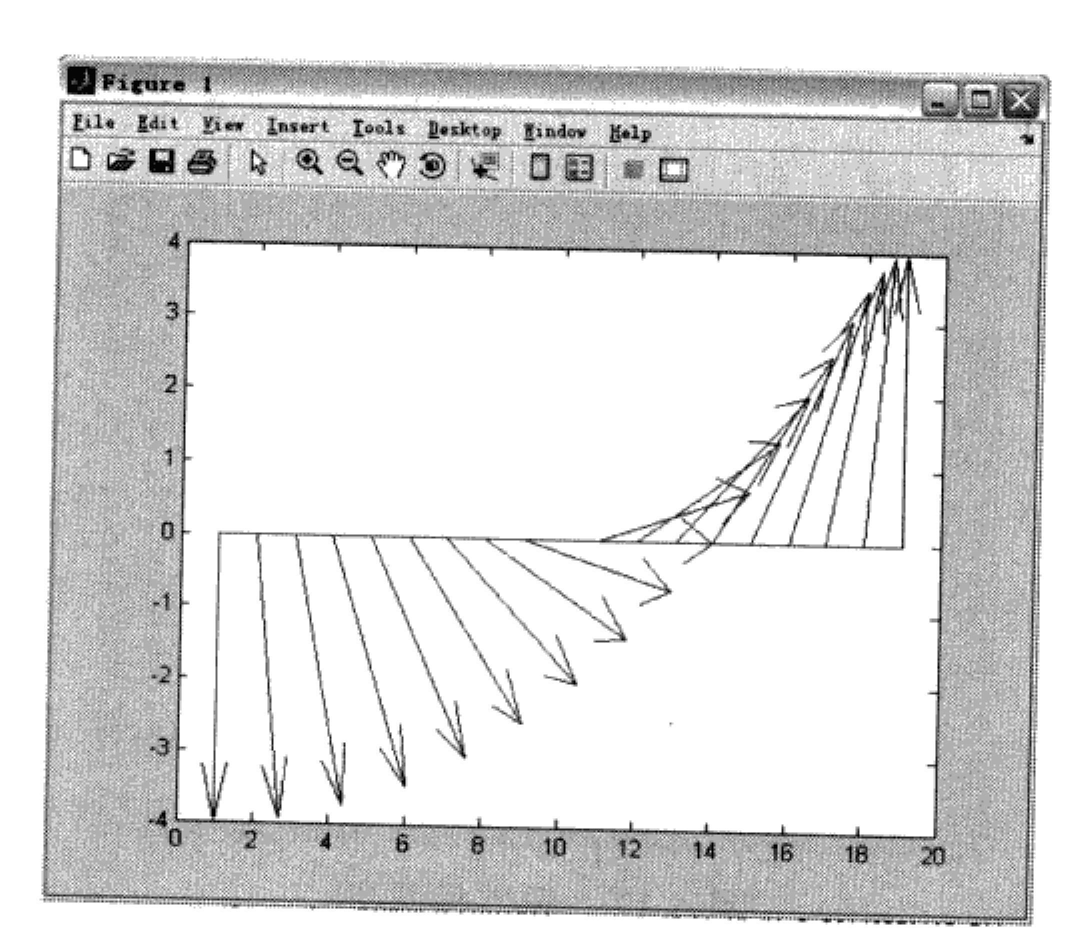


图 6.33　速度向量图

【实例讲解】 u、v 分别包含相对坐标系中的 x 成分和 y 成分。

6.2.15　hist 函数——二维条形直方图

【语法说明】

- n = hist(Y)：把向量 y 中的元素放入等距的 10 个条形中，且返回每一个条形中的元素个数。若 y 为矩阵，则该函数按列对 y 进行处理。
- n = hist(Y,x)：参量 x 为向量，把 y 中元素放到 m（m=length(x)）个由 x 中元素指定的位置为中心的条形中。
- n = hist(Y,nbins)：参量 nbins 为标量，用于指定条形的数目。
- [n,xout] = hist(…)：返回向量 n 与包含频率计数与条形的位置向量 xout，用户可以用函数 bar(xout,n)画出条形直方图。

【功能介绍】 二维条形直方图，可以显示出数据的分配情形。所有向量 y 中的元素或者是矩阵 y 中的列向量中的元素是根据它们的数值范围来分组的，每一组作为一个条形进行显示。条形直方图

中的 x 轴反映了数据 y 中元素数值的范围，直方图的 y 轴显示出参量 y 中的元素落入该组的数目。

【实例 6.33】 绘制二维条形直方图。

```
>>x = -5:0.1:5;
>>y = randn(1000,1);
>>hist(y,x)
```

得到的直方图如图 6.34 所示。

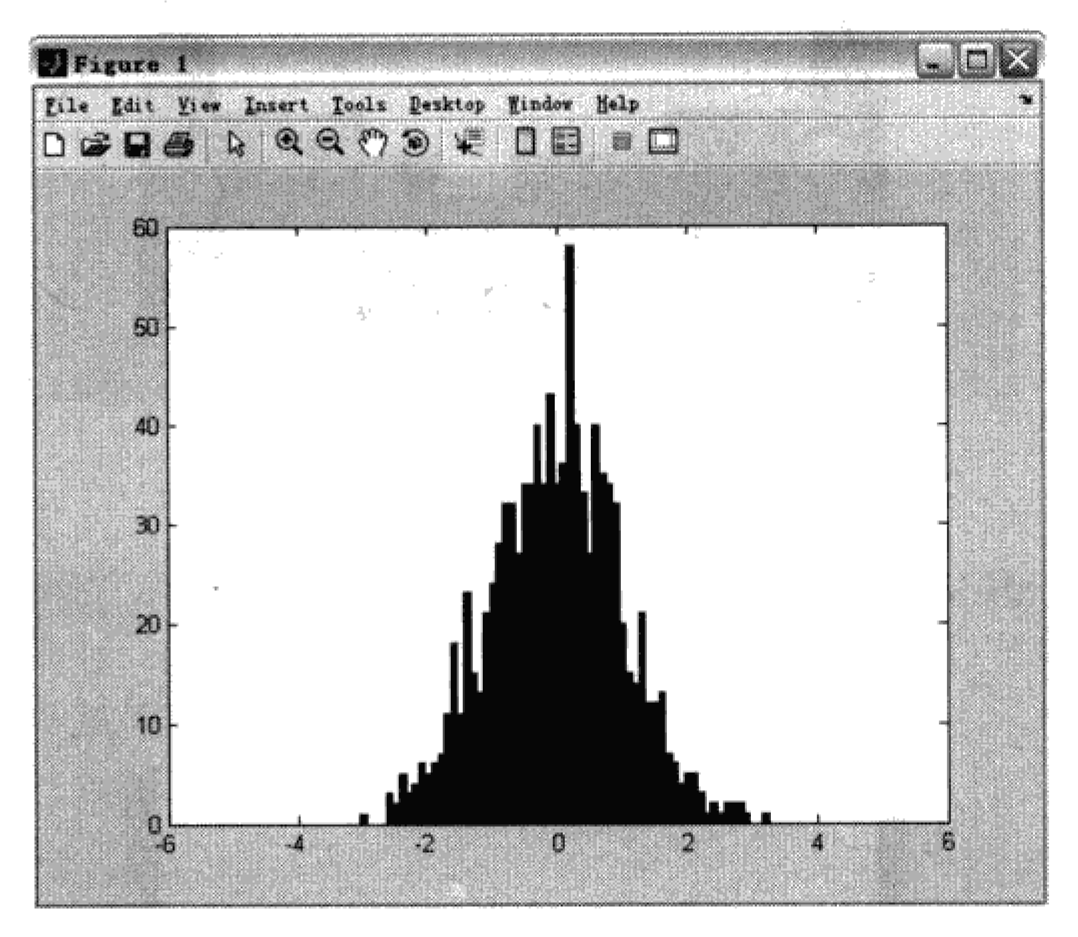


图 6.34　条形直方图

【实例讲解】 用户可以修改直方图的参数，当数值足够大的时候，直方图会很接近正态图。

【实例 6.34】 修改参数，查看直方图的变化。

```
>> x=-3:0.1:3;
>> y=sin(x); % 注意 x 是径度
>> hist(y) % 画出 sin(y)的 histogram,横轴代表 y 的极值[-1,1],纵轴代表 y 的个数
>> hist(y,25) % 将预设 10 个长条改为 25 个,纵轴的值也改变
>> hist(y,x) % 将横轴上下限改为-3 到 3,纵轴的值也改变
```

得到的结果如图 6.35～图 6.37 所示。

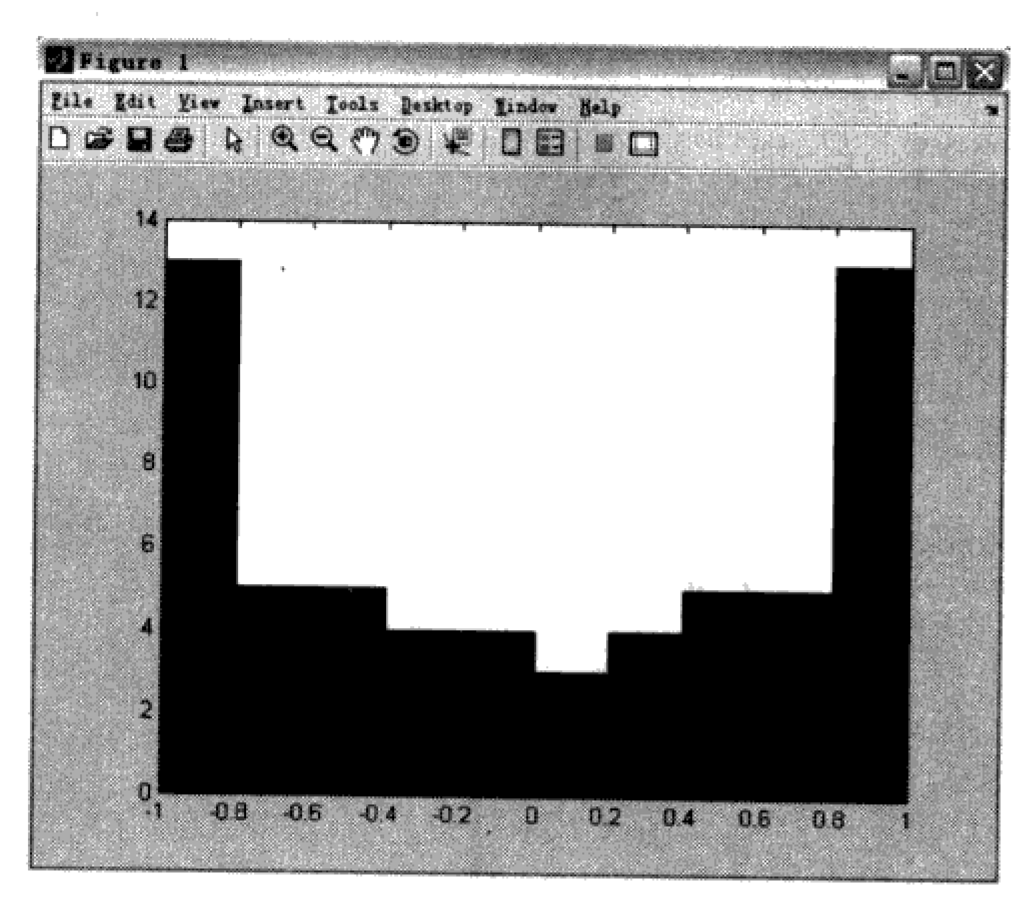


图 6.35 原始图

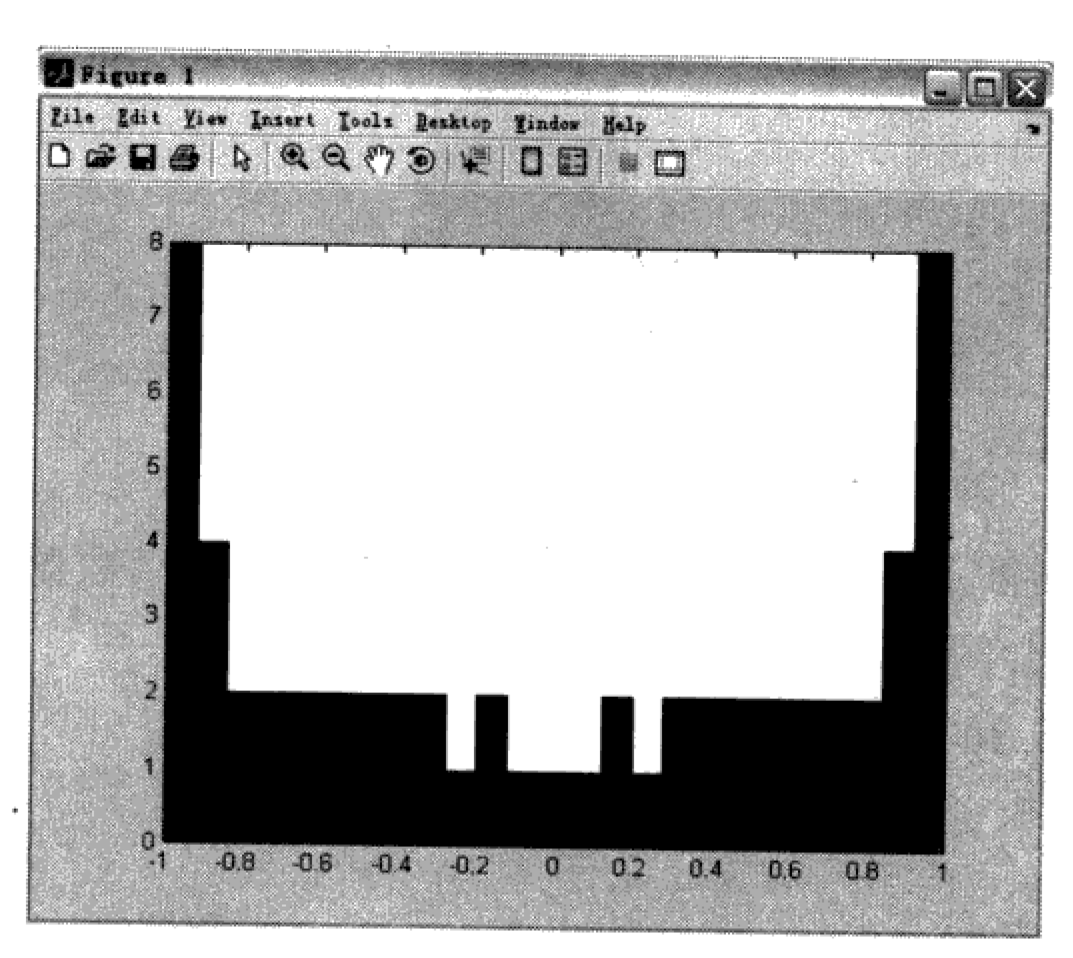


图 6.36 修改后的图表

【实例讲解】 通过长条数目的变化和横轴上下限的变化，观察图形的变化。

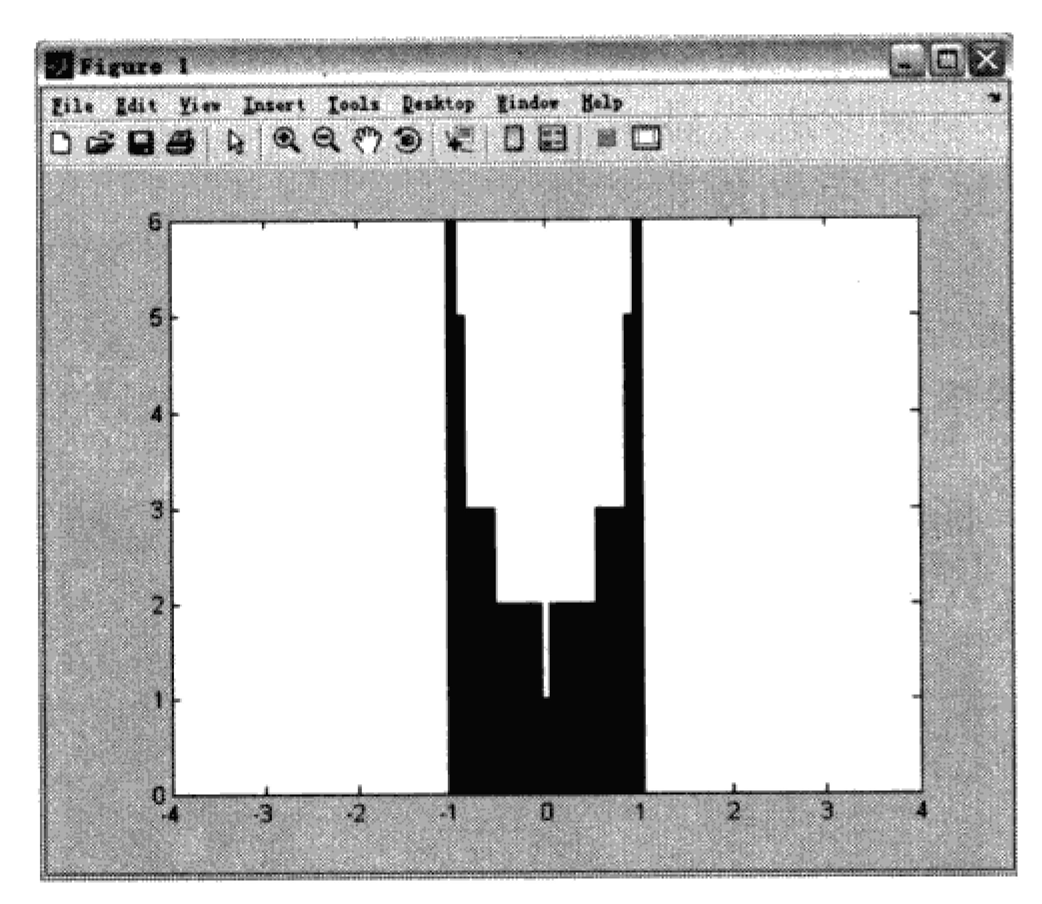


图 6.37 再次修改的图表

6.2.16 rose 函数——角度直方图

【语法说明】

- rose(theta)：画一角度直方图，显示参数 theta 的数据在 20 个区间或更少的区间内的分布。向量 theta 中的角度单位为弧度，用于确定每一区间与原点的角度。每一区间的长度反映出输入参量的元素落入一区间的个数。
- rose(theta,x)：用参量 x 指定每一区间内的元素与区间的位置，length(x)等于每一区间内元素的个数与每一区间位置角度的中间角度。例如，若 x 为一 5 维向量，rose 函数分配参量 theta 中的元素为 5 部分，每一部分的角度中线由 x 指定。
- rose(theta,nbins)：于区间 [0,2*pi] 内画出 nbins 个等距的小扇形。默认值为 20。
- [tout,rout] = rose(…)：返回向量 tout 与 rout，可以用 polar(tout, rout)画出图形。

【功能介绍】 画角度直方图。该直方图是一个显示所给数据的

变化范围内数据的分布情形的极坐标图。所给数据分成不同的组，每一组作为一小扇形进行显示。

【实例 6.35】 画角度直方图。

```
>>theta = 3*pi*randn(1,30);
>>rose(theta)
```

得到的图形如图 6.38 所示

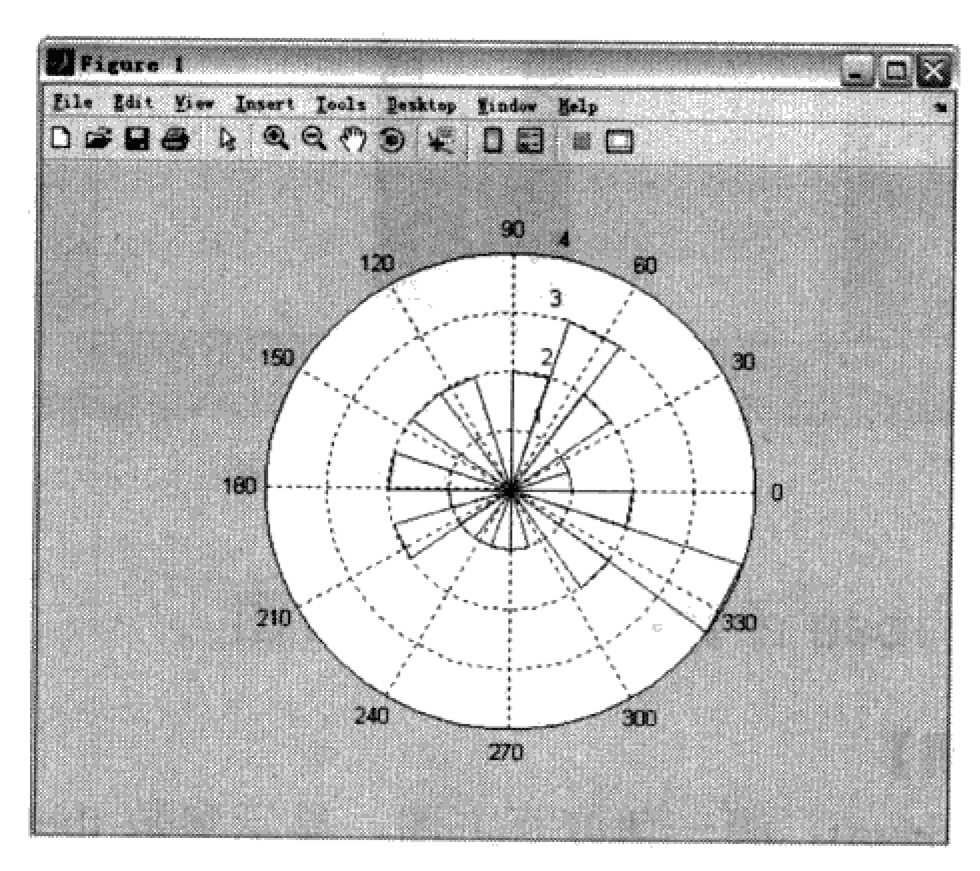


图 6.38 条形直方图

【实例讲解】 在上面的例子中，参数 theta 指定了角度。

6.2.17 stem 函数——画二维离散数据图

【语法说明】

- stem(Y)：按 y 元素的顺序画出柄形图，在 *x* 轴上，柄与柄之间的距离相等；若 y 为矩阵，则把 y 分成几个行向量，在同一横坐标的位置上画出一个行向量的柄图。
- stem(X,Y)：在横坐标 *x* 上画出列向量 y 的柄形图 。其中 x 与 y 为同型的向量或矩阵，此外，x 可以为行向量或列向量，而 y 为有 m=length(x)行的矩阵。
- stem(…,'fill')：指定是否对柄形图末端的小圆圈填充颜色。

- stem(…,LineSpec)：用参数 LineSpec 指定线型，标记符号和柄图末端的小圆圈的颜色画柄图。
- h = stem(…)：返回柄形图的 line 图形对象句柄向量。

【功能介绍】 画二维离散数据图。该图用线条显示数据点与 *x* 轴的距离，在线的末端用一小圆圈（默认标记）或用指定的其他标记 *y* 轴上方向的长度。

【实例 6.36】 画二维离散数据图。

```
>>x = linspace(0,2,10);
>>stem(exp(-x.^2),'fill','-.')
```

得到的图形如图 6.39 所示。

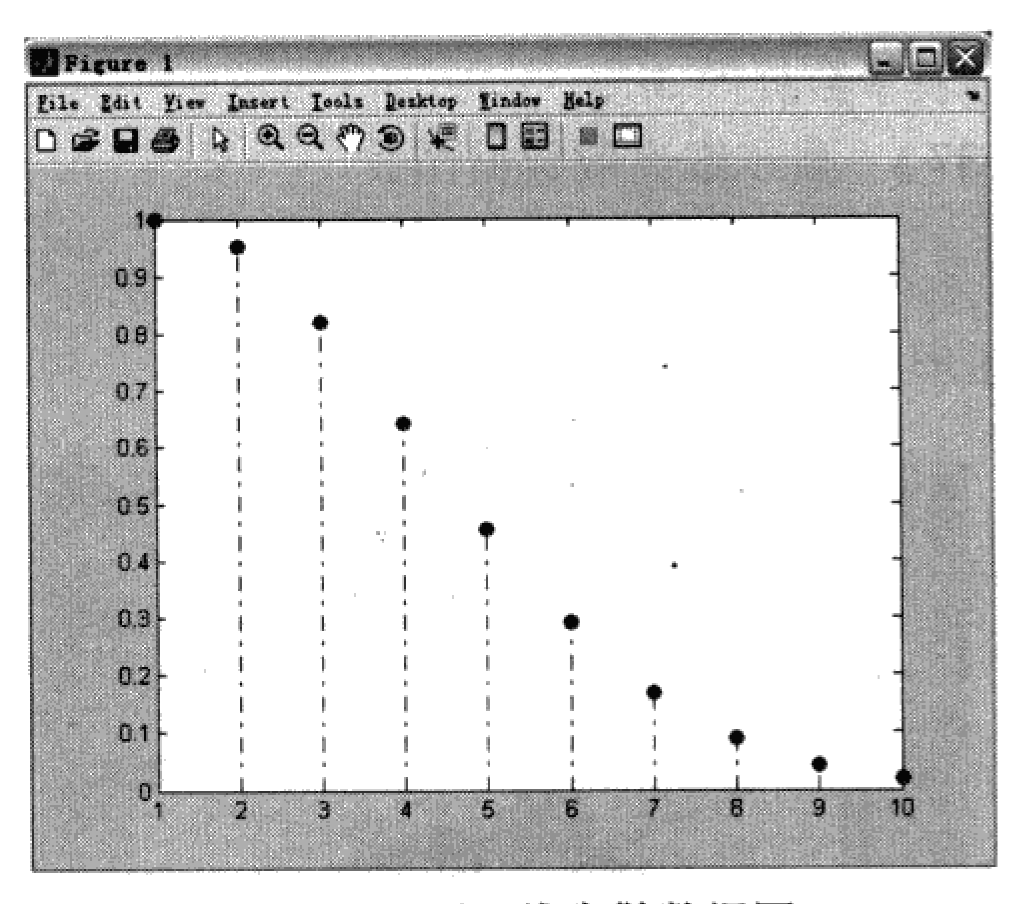


图 6.39　画二维离散数据图

【实例讲解】 从上面的图表中可以看出，二维离散数据可以形象的显示离散特征。

6.2.18　stem3 函数——画三维离散数据图

【语法说明】

- stem3(Z)：用柄形图显示 z 中数据与 xy 平面的高度。若 z 为一行向量，则 x 与 y 将自动生成，stem3 将在与 *x* 轴平行的方向

上等距的位置上画出 z 的元素；若 y 为列向量，stem3 将在与 y 轴平行的方向上等距的位置上画出 z 的元素。

- stem3(X,Y,Z)：在参数 x 与 y 指定的位置上画出 z 的元素，其中 x、y、z 必须为同型的向量或矩阵。
- stem3(…,'fill')：指定是否要填充柄形图末端小圆圈。
- stem3(…,LineSpec)：指定线型，标记符号和末端小圆圈的颜色。
- h = stem3(…)：返回柄形图的 line 图形对象句柄。

【功能介绍】 画三维离散数据的柄形图。该图用一线段显示数据离开 *xy* 平面的高度，在线段的末端用一小圆圈（默认记号）或其他的标记符号表示数据的高度。

【实例 6.37】 画三维离散数据图。

```
[X,Y,Z] = peaks(20);
stem3(X,Y,Z, 'r*')
```

得到的图形如图 6.40 所示。

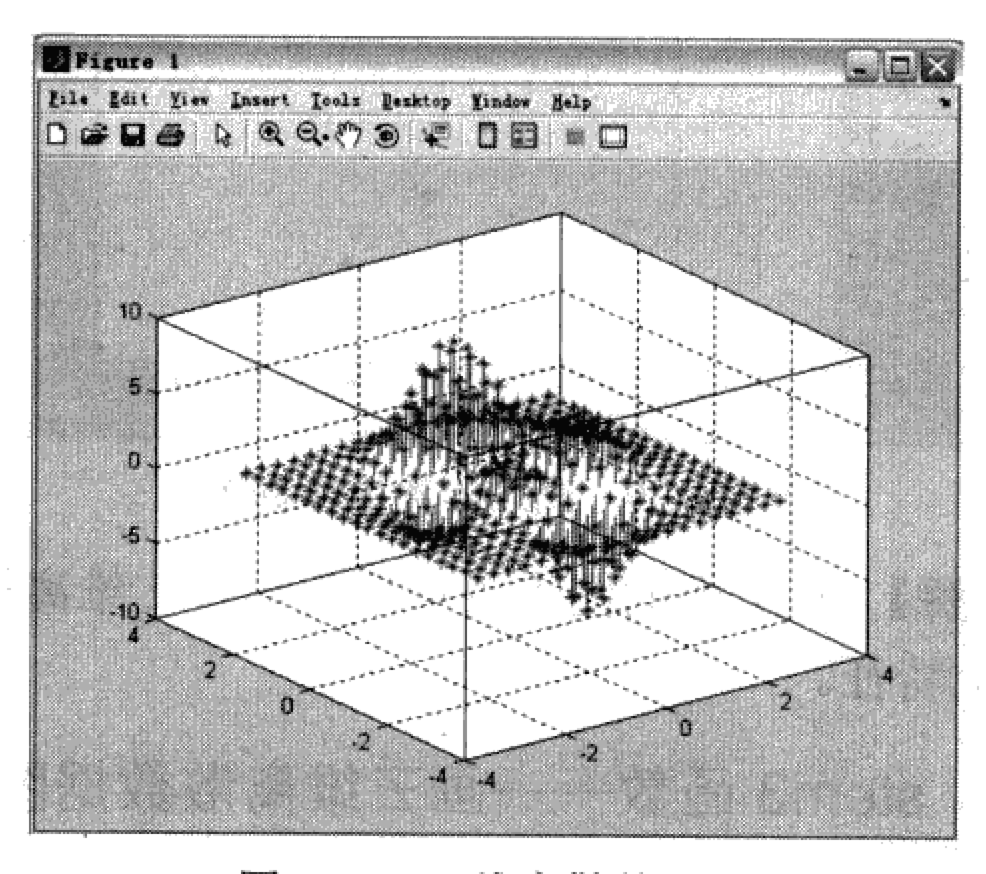


图 6.40　三维离散数据图

【实例讲解】 peaks 函数为多峰函数，常用在等高线绘制的演示中。

6.2.19　pie 函数——画饼图

【语法说明】

- pie(X)：用 x 中的数据画一饼形图，x 中的每一元素代表饼形图中的一部分。X 中元素 X(i)所代表的扇形大小通过 X(i)/sum(X)的大小来决定。若有 sum(X)=1，则 x 中元素就直接指定了所在部分的大小；若 sum(X)<1，则画出一不完整的饼形图。
- pie(X,explode)：从饼形图中分离出一部分，explode 为元素为零或非零的、与 x 相对应的向量或矩阵。与 explode 的非零值对应的部分将从饼形图中心分离出来。explode 必须与 x 同型。
- h = pie(…)：返回一 patch 与 text 的图形对象句柄向量 h。

【功能介绍】 画饼形图。

【实例 6.38】 画饼形图。

```
>>x = [1 3 0.5 2.5 2];
>>explode = [0 1 0 0 0];
>>pie(x,explode)
```

得到的图形如图 6.41 所示。

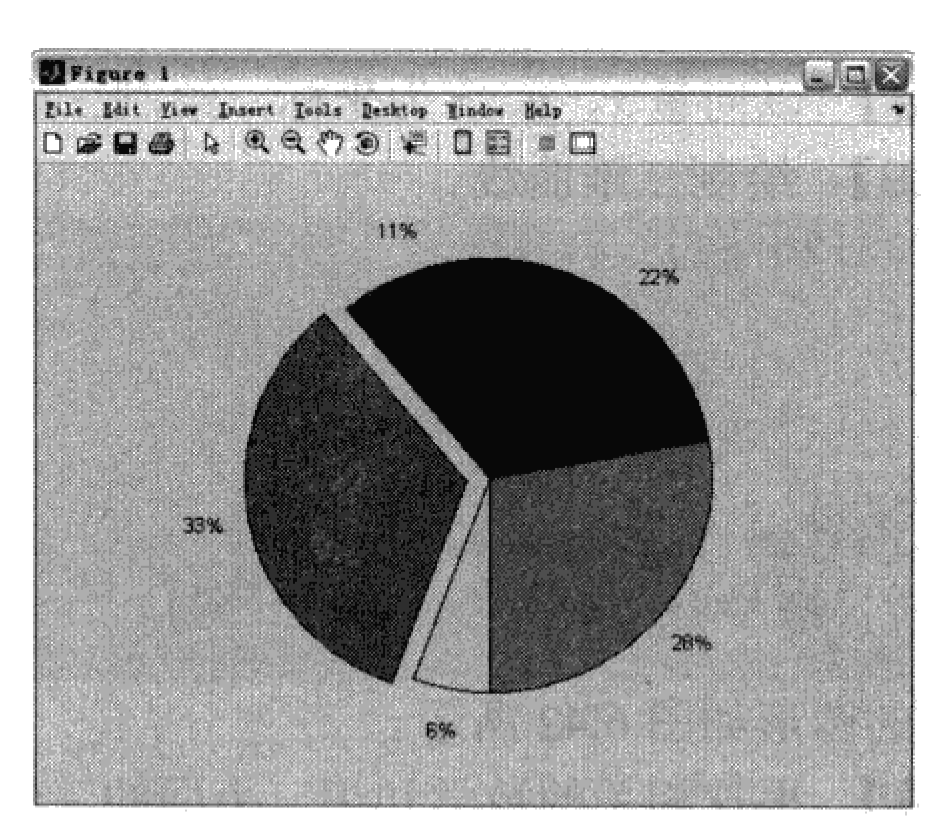


图 6.41　饼形图

【实例讲解】 x = [1 3 0.5 2.5 2] 中的每一元素代表饼形图中的

一部分。

6.3 三维曲线绘制

在工程计算和处理中，通常需要表达 3 个量之间的关系，并且需要以直观的方式表达，使观察者能够一目了然地看到数据量之间的变化趋势。这时，就需要绘制三维曲线图来显示曲线之间的变化规律，下面的小节将要讲解绘制三维曲线的有关函数。

6.3.1 plot3 函数——绘制三维曲线

【语法说明】 plot3(x1,y1,z1,选项 1,x2,y2,z2,选项 2,…,xn,yn,zn,选项 n)。

其中每一组 x,y,z 组成一组曲线的坐标参数，选项的定义和 plot 函数相同。当 x,y,z 是同维向量时，则 x,y,z 对应元素构成一条三维曲线；当 x,y,z 是同维矩阵时，则以 x,y,z 对应列元素绘制三维曲线，曲线条数等于矩阵列数。

【功能介绍】 以向量 *x*、*y*、*z* 为坐标，绘制三维曲线。

【实例 6.39】 绘制三维曲线。

```
t=0:pi/100:20*pi;
x=sin(t);
y=cos(t);
z=t.*sin(t).*cos(t);
plot3(x,y,z);
title('Line in 3-D Space');
xlabel('X');ylabel('Y');zlabel('Z');
grid on;
```

最后得到的图形如图 6.42 所示。

【实例讲解】 根据相关的数学知识，上面的三维曲线图是部分的螺旋线。

【实例 6.40】 绘制三维螺旋曲线。

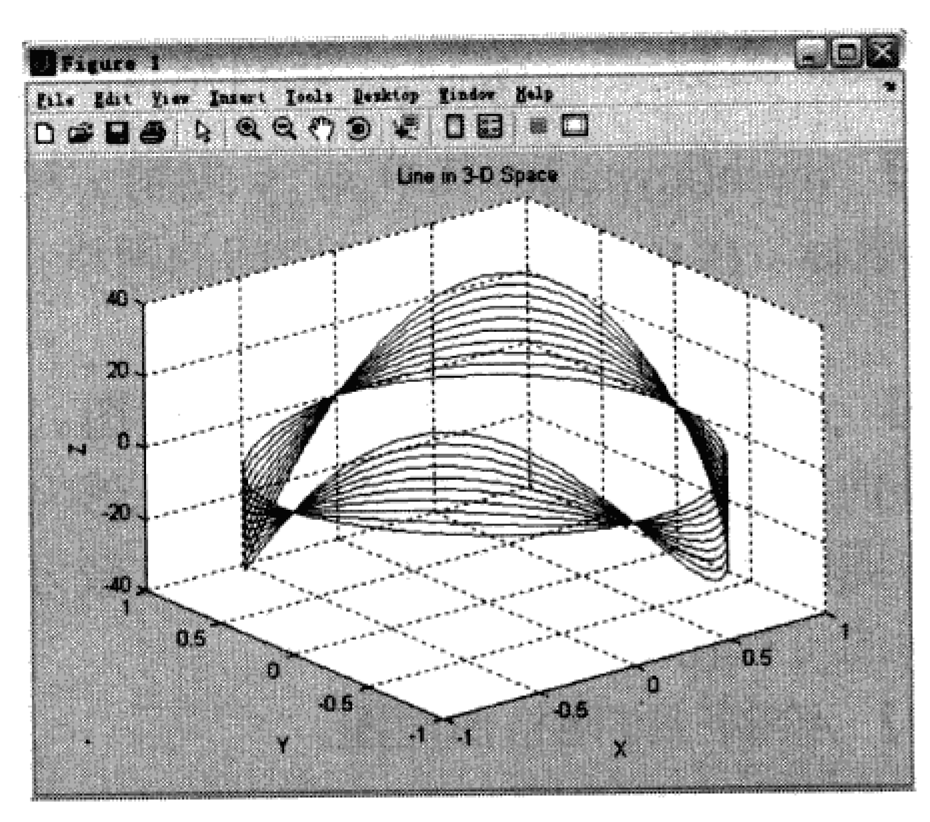


图 6.42 三维曲线图

```
t=0:pi/50:10*pi;
y1=sin(t),y2=cos(t);
plot3(y1,y2,t);
title('helix'),text(0,0,0,'origin');
xlabel('sin(t)'),ylabel('cos(t)'),zlabel('t');
grid;
```

最后得到的图形如图 6.43 所示。

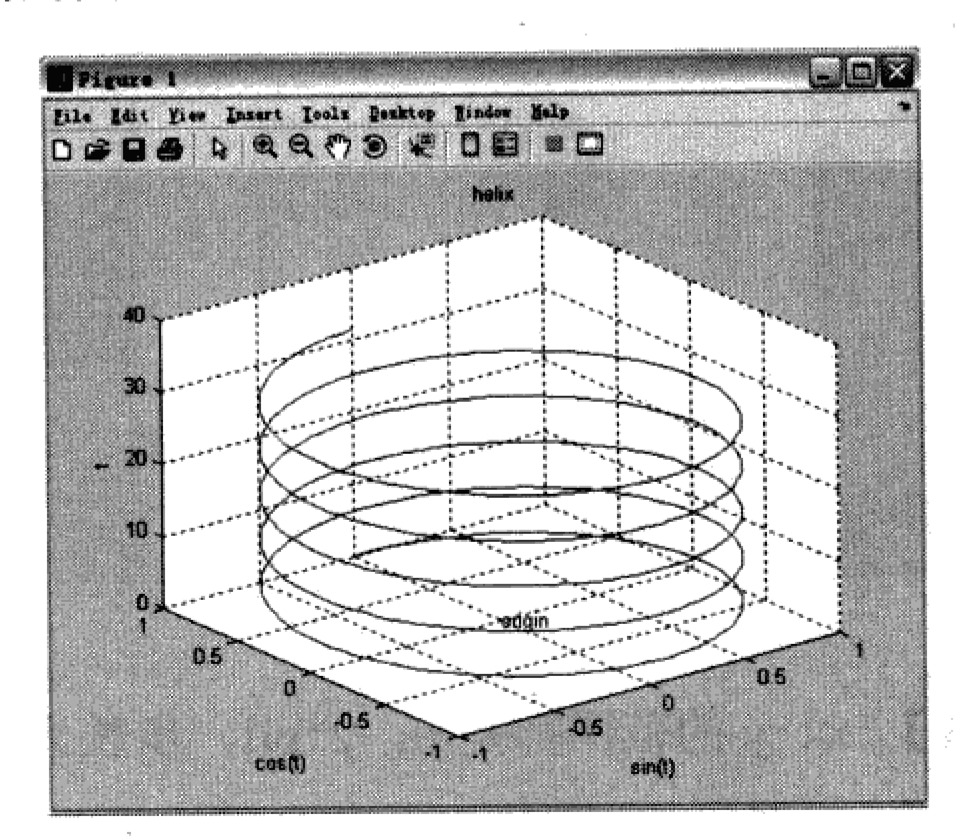


图 6.43 三维螺旋曲线

【实例讲解】 本例中 y1、y2 都是 t 的参数方程。grid 相当于 grid on。

6.3.2 mesh 函数——绘制三维网格图

【语法说明】 mesh(x,y,z,c)：其中 x，y 控制 x 和 y 轴坐标，由(x，y)求得 z 轴坐标，(x,y,z)组成了三维空间的网格点；c 用于控制网格点颜色。

【功能介绍】 mesh 函数用于绘制三维网格图。在不需要绘制特别精细的三维曲面结构图时，可以通过绘制三维网格图来表示三维曲面。三维曲面的网格图最突出的优点是：它较好地解决了实验数据在三维空间的可视化问题。

【实例 6.41】 绘制三维网格曲面图。

```
x=[0:0.15:2*pi];
y=[0:0.15:2*pi];
z=sin(y')*cos(x); %矩阵相乘
mesh(x,y,z);
```

得到的三维网格曲面图如图 6.44 所示。

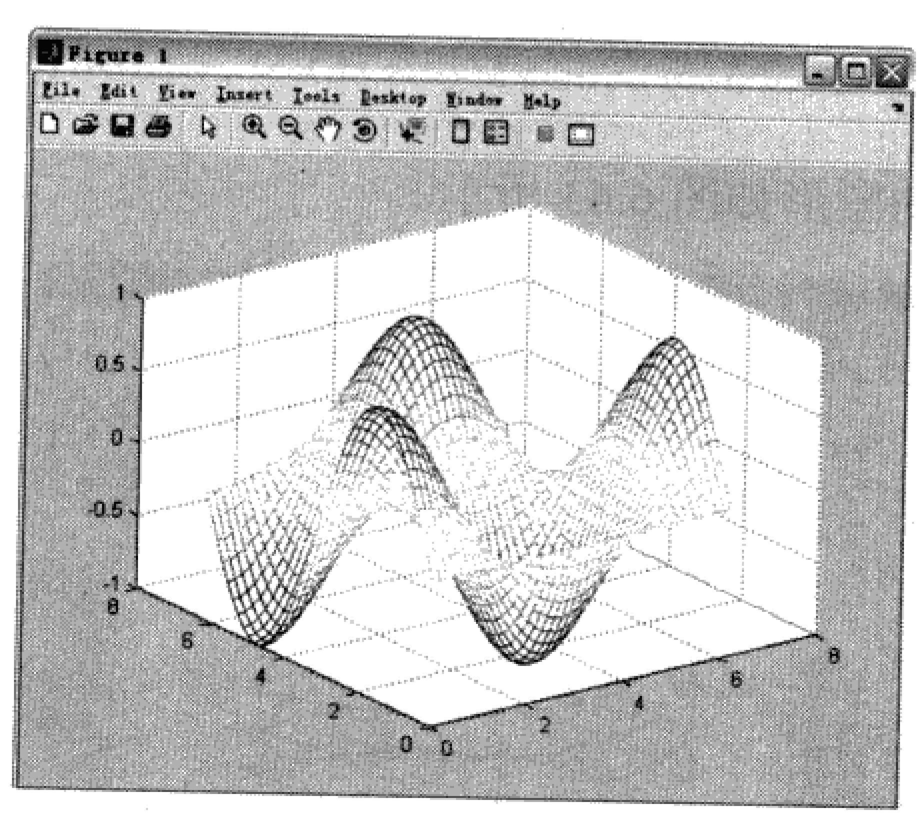


图 6.44 网格曲面图

【实例讲解】 x、y 的范围都是[0，2*pi]，步长为 0.15。

6.3.3 surf 函数——三维曲面图

【语法说明】 surf (x,y,z)：其中 x，y 控制 x 和 y 轴坐标，矩阵

z 是由 x，y 求得的曲面上 z 轴坐标。

【功能介绍】 用于绘制三维曲面图，各线条之间的补面用颜色填充。surf 函数和 mesh 函数的调用格式一致。

【实例 6.42】 绘制三维曲面图形。

```
x=[0:0.15:2*pi];
y=[0:0.15:2*pi];
z=sin(y')*cos(x); %矩阵相乘
surf(x,y,z);
xlabel('x-axis'),ylabel('y-axis'),zlabel('z-label');
title('3-D surf');
```

得到的图形如图 6.45 所示。

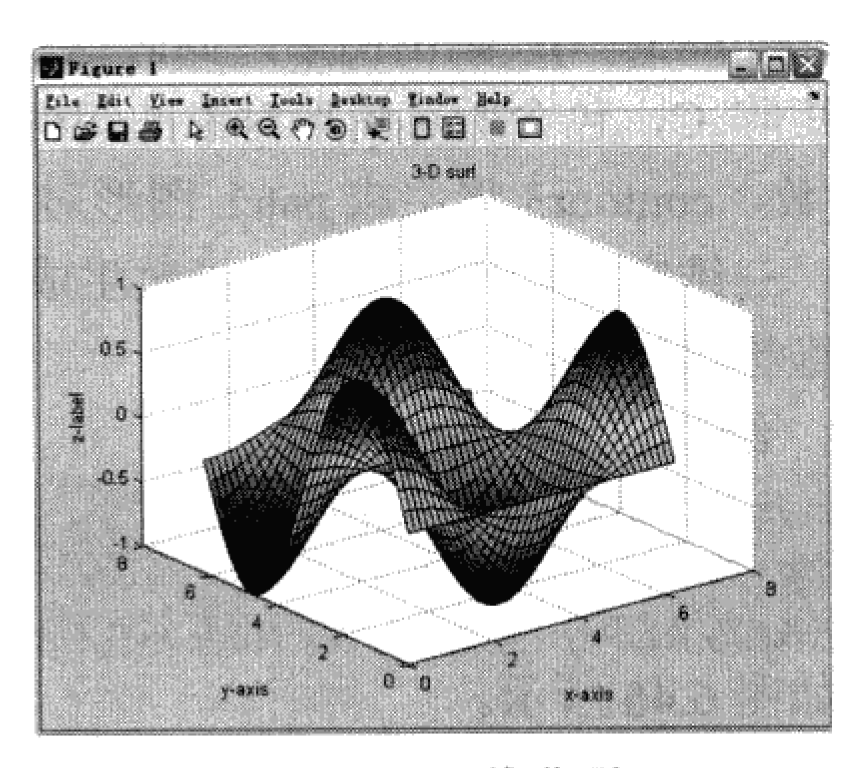


图 6.45　三维曲面

【实例讲解】 用户可以修改数据数组的间隔，来查看的三维曲面。

6.3.4　contour3 函数——三维等高线绘制

【语法说明】

- contour3(Z)：画出三维空间角度观看矩阵 Z 的等高线图，其中 Z 的元素被认为是距离 xy 平面的高度，矩阵 Z 至少为 2×2 阶的。等高线的条数与高度是自动选择的。
- contour3(Z,n)：画出由矩阵 Z 确定的 n 条等高线的三维图。
- contour3(Z,v)：在参量 v 指定的高度上画出三维等高线，当

然等高线条数与向量 v 的维数相同。

contour3(X,Y,Z)、contour3(X,Y,Z,n)、contour3(X,Y,Z,v)：用 X 与 Y 定义 x 轴与 y 轴的范围。若 X 为矩阵，则 X(1,:)定义 x 轴的范围；若 Y 为矩阵，则 Y(:,1)定义 y 轴的范围；若 X 与 Y 同时为矩阵，则它们必须同型。不论为哪种使用形式，所起的作用与函数 surf 相同。若 X 或 Y 有不规则的间距，contour3 还是使用规则的间距计算等高线，然后将数据转变给 X 或 Y。

- contour3(…,LineSpec)：用参量 LineSpec 指定的线型与颜色画等高线。
- [C,h] = contour3(…)：画出图形，同时返回与函数 contourc 中相同的等高线矩阵 C，包含所有图形对象的句柄向量 h；若没有指定 LineSpec 参数，contour3 将生成 patch 图形对象。

【功能介绍】 三维空间等高线图。该函数生成一个定义在矩形格栅上曲面的三维等高线图。

【实例 6.43】 绘制三维等高线。

```
>>[X,Y] = meshgrid([-4:.2:4]);
>>Z = X.*exp(-X.^2-Y.^2);
>>contour3(X,Y,Z,30)
```

得到的图形如图 6.46 所示。

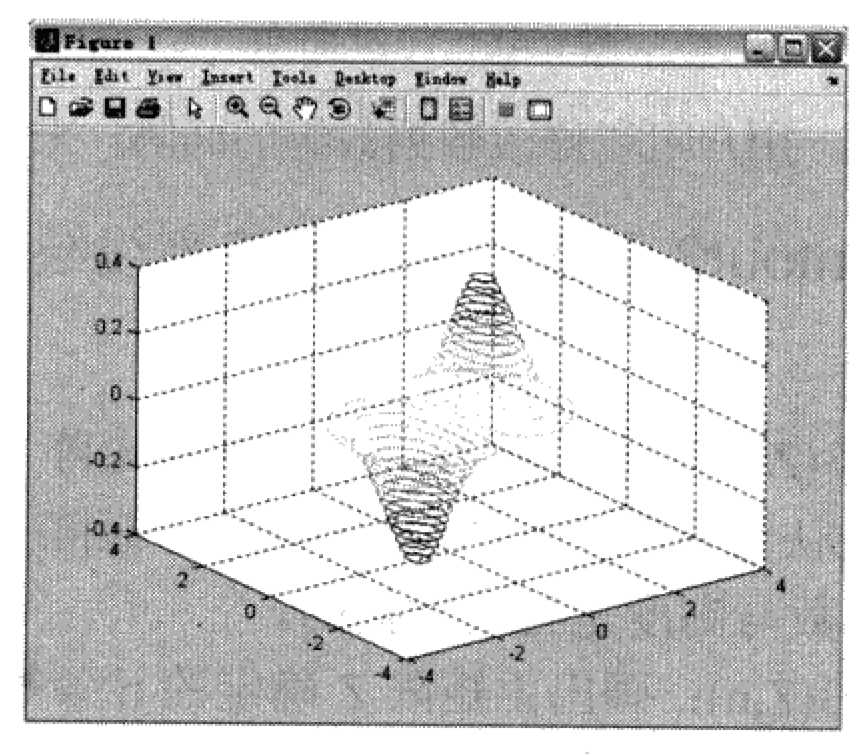


图 6.46 三维等高线

【实例讲解】 用户可以将这个函数和后面的contour 函数进行对比。

6.3.5 contour 函数——曲面的等高线

【语法说明】

- contour(z)：把矩阵 z 中的值作为一个二维函数的值，等高曲线是一个平面的曲线，平面的高度 v 是 Matlab 自动取的。
- contour(x,y,z)：(x,y)是平面 z=0 上点的坐标矩阵，z 为相应点的高度值矩阵。效果同上。
- contour(z,n)：画出 *n* 条等高线。
- contour(x,y,z,n)：画出 *n* 条等高线。
- contour(z,v)：在指定的高度 v 上画出等高线。
- contour(x,y,z,v)：在指定高度 v 上画等高线。

【功能介绍】 绘制曲面的等高线图。

【实例 6.44】 绘制曲面的等高线图。

```
>>contour(peaks(40))
```

得到的图形如图 6.47 所示。

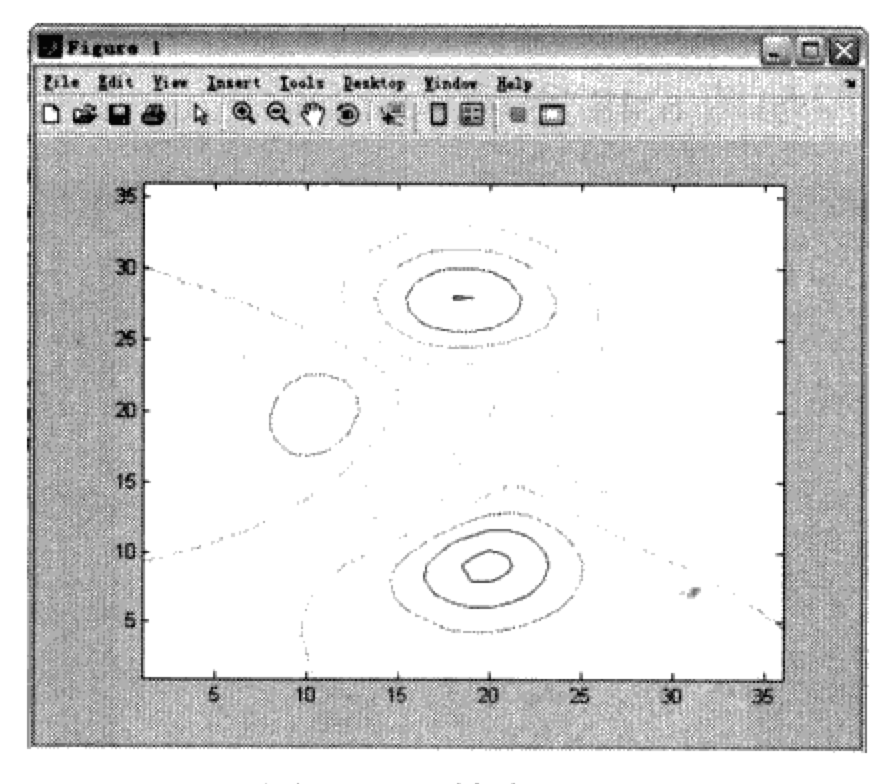


图 6.47　等高线图

【实例讲解】 在上面的函数中，首先使用 peaks 函数绘制三维图形，然后使用 contour 函数绘制等高线。

6.3.6 clabel 函数——等高线填标签

【语法说明】

- clabel(C,h)：把标签旋转到恰当的角度，再插入到等高线中。只有等高线之间有足够的空间时才加入，当然这决定于等高线的尺度。
- clabel(C,h,v)：在指定的高度 v 上显示标签 h，并对标签做恰当的处理。
- clabel(C,h,'manual')：手动设置标签。用户用鼠标左键或空格键在最接近指定的位置上放置标签，用键盘上的回车键结束该操作。
- clabel(C)：在从函数 contour 生成的等高线结构 C 的位置上添加标签。此时标签的放置的位置是随机的。
- clabel(C,v)：在给定的位置 v 上显示标签。
- clabel(C,'manual')：允许用户通过鼠标来给等高线贴标签。

【功能介绍】 在二维等高线图中添加高度标签。

【实例 6.45】 给等高线作标注。

```
>>[x,y] = meshgrid(-3:.2:3);
>>z = x.*y.*exp(-x.^2-y.^2);
>>[C,h] = contour(x,y,z);
>>clabel(C,h);
```

得到的结果图如图 6.48 所示。

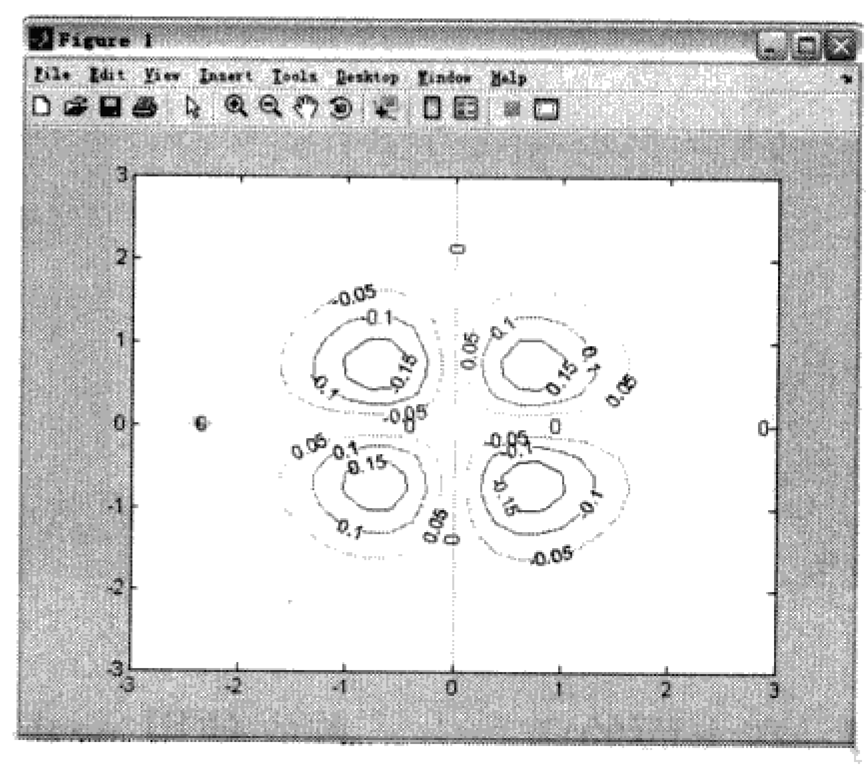


图 6.48 为等高线作标记

【实例讲解】 h 为标签，在图中显示在适当的位置。

6.3.7　contourc 函数——等高线图形计算

【语法说明】

- C = contourc(Z)：从矩阵 Z 中计算等高矩阵，其中 Z 的维数至少为 2×2 阶，等高线为矩阵 Z 中数值相等的单元。等高线的数目和相应的高度值是自动选择的。
- C = contourc(Z,n)：在矩阵 Z 中计算出 *n* 个高度的等高线。
- C = contourc(Z,v)：在矩阵 Z 中计算出给定高度向量 v 上计算等高线，当然向量 v 的维数决定了等高线的数目。若只要计算一条高度为 a 的等高线，输入 contourc(Z,[a,a])。
- C = contourc(x,y,Z)：在矩阵 Z 中，参量 x/y 确定的坐标轴范围内计算等高线。
- C = contourc(x,y,Z,n)：从矩阵 Z 中，参量 x 与 y 确定的坐标范围内画出 *n* 条等高线。
- C = contourc(x,y,Z,v)：从矩阵 Z 中，参量 x 与 y 确定的坐标范围内，画出 v 指定的高度上的等高线。

【功能介绍】 低级等高线图形计算函数。该函数计算等高线矩阵 *c*，该矩阵可用于函数 contour、contour3 和 contourf 等。矩阵 *Z* 中的数值确定平面上的等高线高度值，等高线的计算结果用由矩阵 *Z* 维数决定的间隔的宽度。

【实例 6.46】 从矩阵 *Z* 中计算等高矩阵。

```
>> [x,y] = meshgrid(-3:.2:3);
>> z = x.*y;
>> C = contourc(z)

C =

  Columns 1 through 8

   -8.0000 2.6667 2.0000 1.7143 1.0000 -8.0000 29.3333
30.0000
```

```
        4.0000 31.0000 30.2857 30.0000 29.3333 4.0000 1.0000
1.7143

    Columns 9 through 16
  ............
```

【实例讲解】 上面列出了矩阵 *z* 中所有的等高矩阵，后面省略了一部分。

6.3.8 fill3 函数——填充三维图

【语法说明】

- fill3(X,Y,Z,C)：填充由参数 x、y 和 z 确定的多边形。若 x，y 或 z 为矩阵，fill3 生成 *n* 个多边形，其中 n 为矩阵的列数。在必要的时候，fill3 会自动连接最后一个节点和第一个节点。以便能形成封闭的多边形。参数 c 指定颜色，这儿 c 为引用当前色图的下标向量或矩阵。若 c 为行向量，则 c 的维数必须等于 x 的列数和 y 的列数，若 c 为列向量，则 c 的维数必须等于矩阵 x 的行数和 y 的行数。
- fill3(X,Y,Z,ColorSpec)：用指定的颜色 ColorSpec 填充由 x、y 和 z 确定的多边形。
- fill3(X1,Y1,Z1,C1,X2,Y2,Z2,C2,…)：对多边形的不同区域用不同的颜色进行填充。
- fill3(…,'PropertyName',PropertyValue)：允许用户对特定的 patch 属性进行设置。
- h = fill3(…)：返回 patch 图形对象的句柄向量，每一块（patch）对应一个句柄。

【功能介绍】 用指定的颜色填充三维多边形，阴影类型为平面型和 Gouraud 型。

【实例 6.47】 先生成随机向量，然后绘制三维图形并着色。

```
>>X = 10*rand(4);
>>Y=10*rand(4);
>>Z=10*rand(4);
>>C = rand(4);
>>fill3(X,Y,Z,C)
```

这个例子得到的图形如图 6.49 所示。

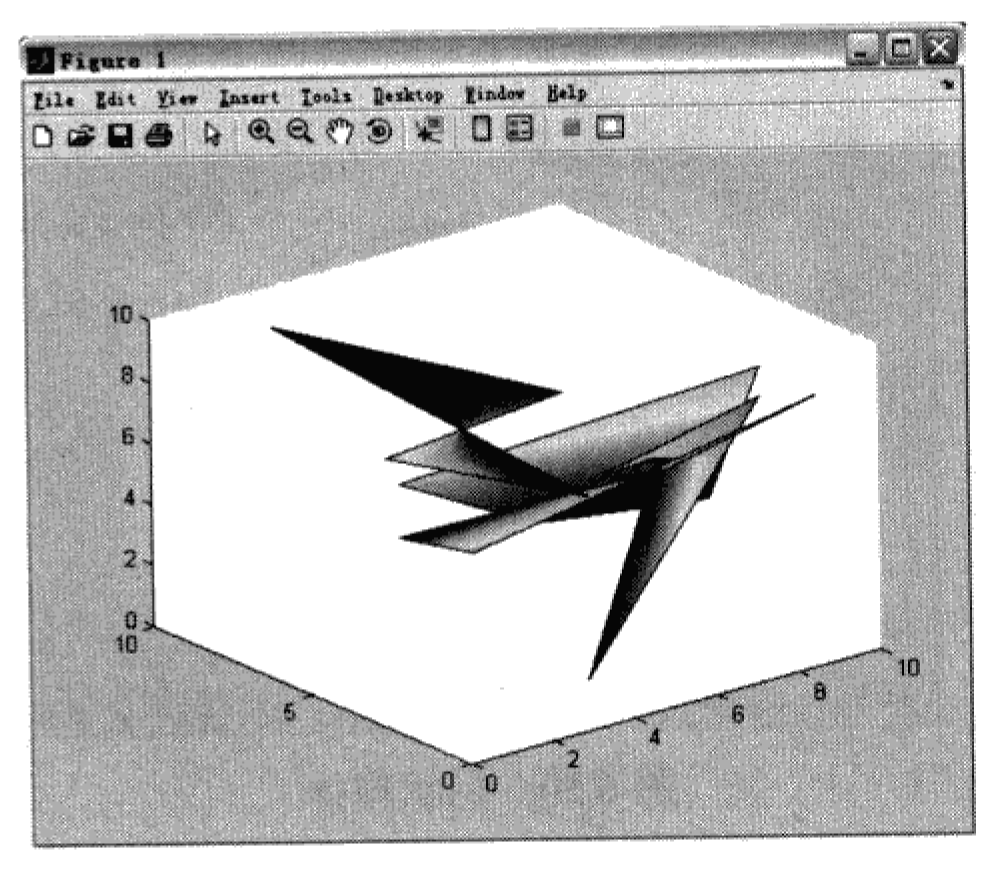


图 6.49　三维图形填充

【实例讲解】 X、Y、Z、C 都是随机得到的数据。再以 X、Y、Z3 个向量为 3 个方向填充图形。

6.3.9　sphere 函数——绘制球体

【语法说明】

- sphere：生成三维直角坐标系中的单位球体，该单位球体由 400 个面组成。
- sphere(n)：在当前坐标系中画出有 $n \times n$ 个面的球体。
- [X,Y,Z] = sphere(n)：返回 3 个阶数为(n+1)*(n+1)的，直角坐标系中的坐标矩阵。该函数没有画图，只是返回矩阵。用户可以用函数 surf（X，Y，Z）或 mesh（X，Y，Z）画出球体。

【功能介绍】 生成球体。

【实例 6.48】 绘制网状球体。

```
>>[X,Y,Z]=sphere;
>>mesh(X,Y,Z)
```

得到的球体如图 6.50 所示。

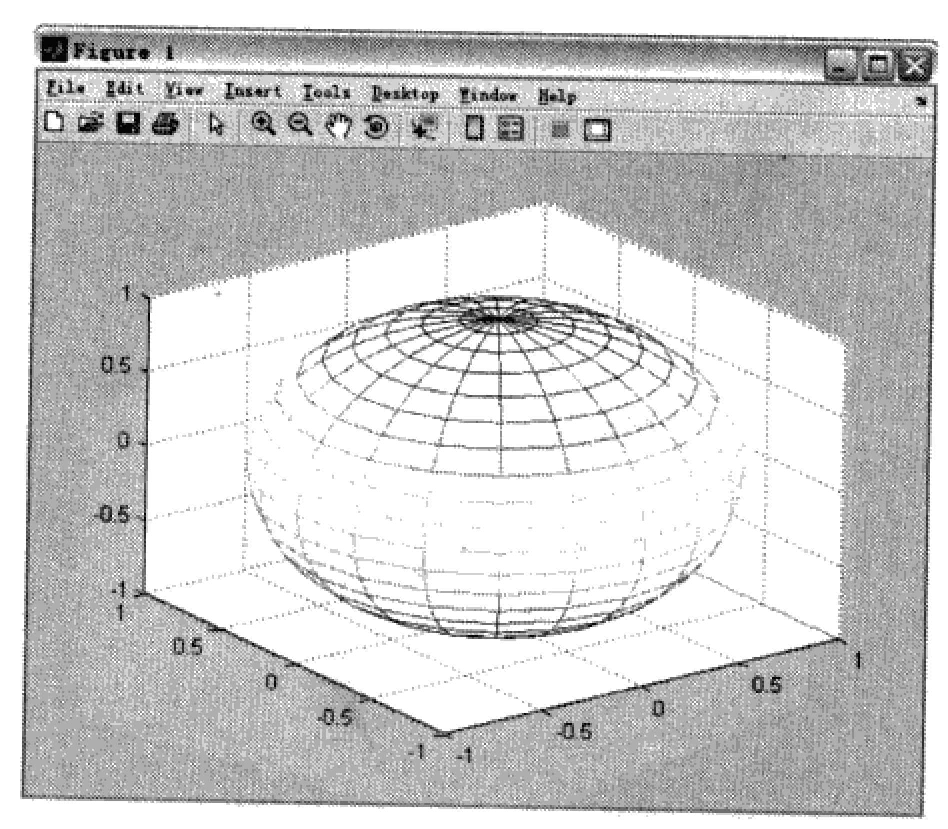


图 6.50 网状球体

【实例讲解】 绘制的是彩色网球图。

6.3.10 contourf 函数——填充二维等高线

【语法说明】

- contourf(Z)：矩阵 Z 的等高线图，其中 Z 理解成距平面的高度。Z 至少为 2×2 阶。等高线的条数与高度是自动选择的。
- contourf(Z,n)：画出矩阵 Z 的 *n* 条高度不同的等高线。
- contourf(Z,v)：画出矩阵 Z 的、由 v 指定的高度的等高线图。
- contourf(X,Y,Z)、contourf(X,Y,Z,n)、contourf(X,Y,Z,v)：画出矩阵 Z 的等高线图，其中 X 与 Y 用于指定 *x* 轴与 *y* 轴的范围。若 X 与 Y 为矩阵，则必须与 Z 同型。
- [C,h,CF] = contourf(…)：画出图形，同时返回与函数 contourc 中相同的等高线矩阵 C，C 也可被函数 clabel 使用；返回包含 patch 图形对象的句柄向量 h；返回一用于填充用的矩阵 CF。

【功能介绍】 填充二维等高线图。即先画出不同等高线，然后相邻的等高线之间用同一颜色进行填充。填充用的颜色决定于当前的色图颜色。

【实例 6.49】 为等高线填色。

```
>>contourf(peaks(40),30);
>>colormap gray
```

当执行第一个函数时得到图 6.51 所示，当两个函数执行后得到图 6.52 所示的结果。

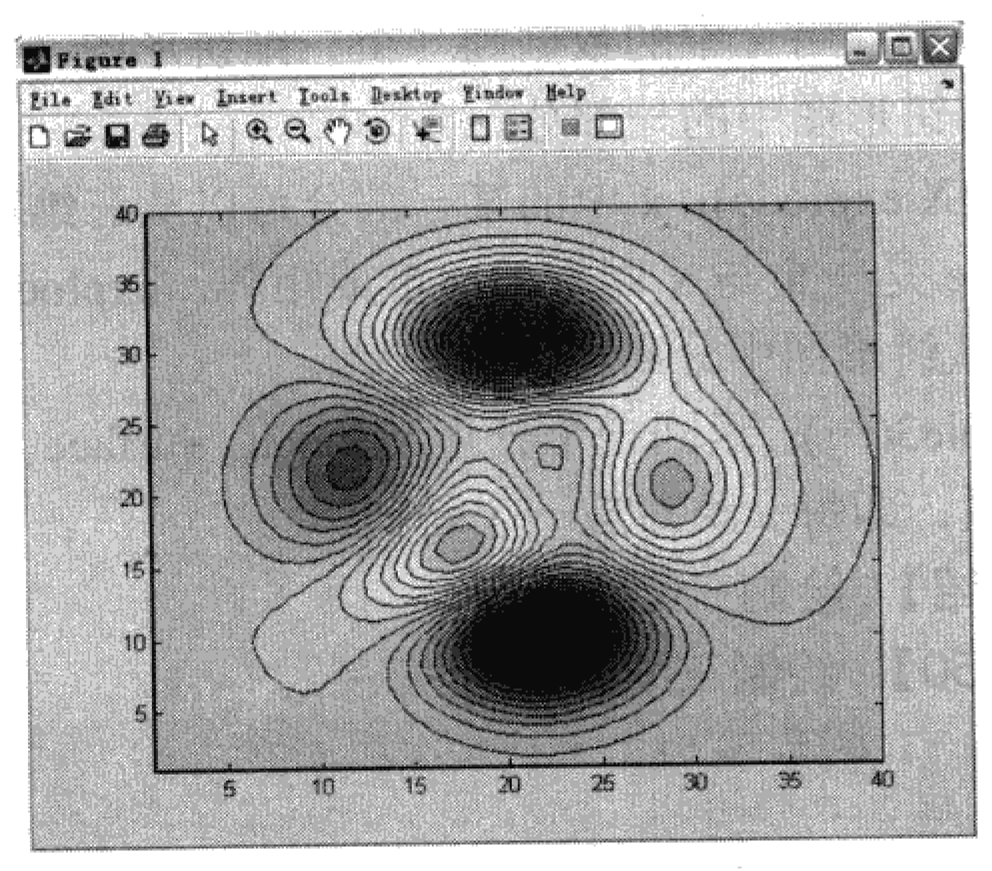


图 6.51 等高线填自然色

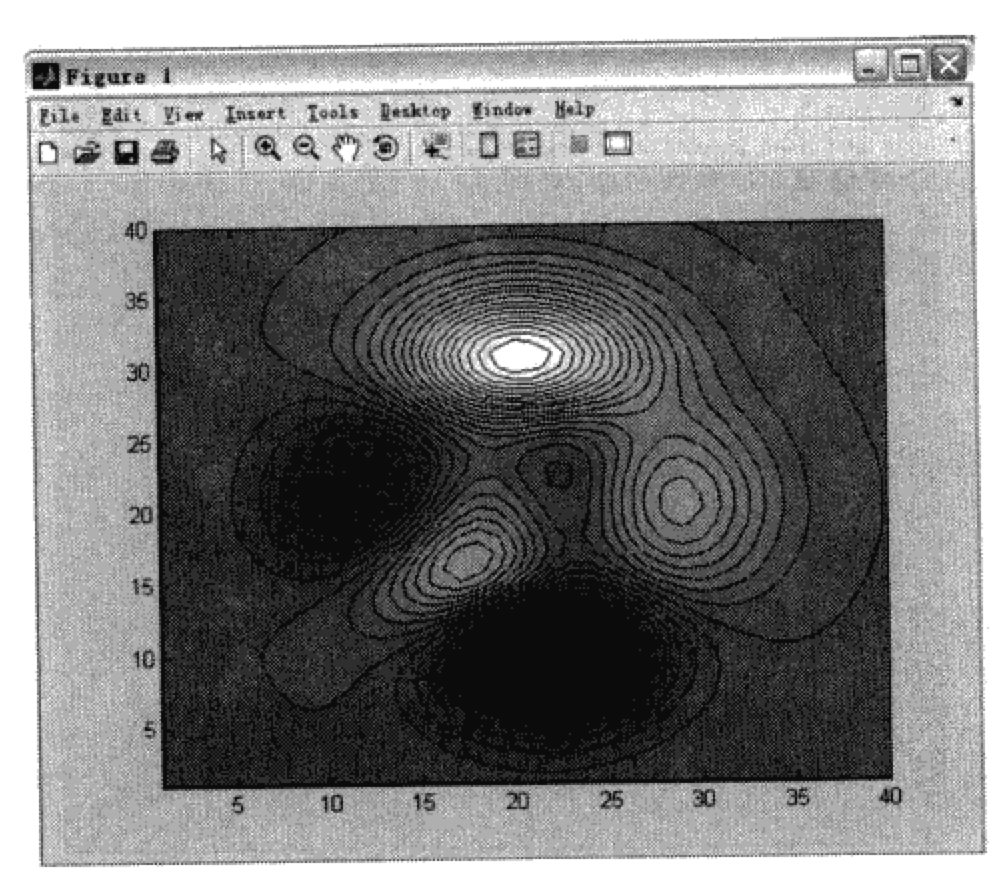


图 6.52 等高线填灰色

【实例讲解】 从上面的结果中可以看出，用灰色填充等高线的时候，将用亮度来区别数值。

6.3.11 pie3 函数——三维饼图

【语法说明】

- pie3(X)：用 x 中的数据画一个三维饼形图。X 中的每一个元素代表三维饼形图中的一部分。
- pie3(X,explode)：x 中的某一部分可以从三维饼形图中分离出来。explode 是一个与 x 同型的向量或矩阵，explode 中非零的元素对应 X 中从饼形图中分离出来的分量。
- h = pie3(…)：返回一个分量为 patch，surface 和 text 图形句柄对象的向量。即每一块对应一个句柄。

【功能介绍】 绘制三维饼形图。

【实例 6.50】 绘制三维饼形图。

```
>>x = [1 3 0.5 2.5 2]
>>ex = [0 1 0 0 0]
>>pie3(x,ex)
```

得到的结果如图 6.53 所示。

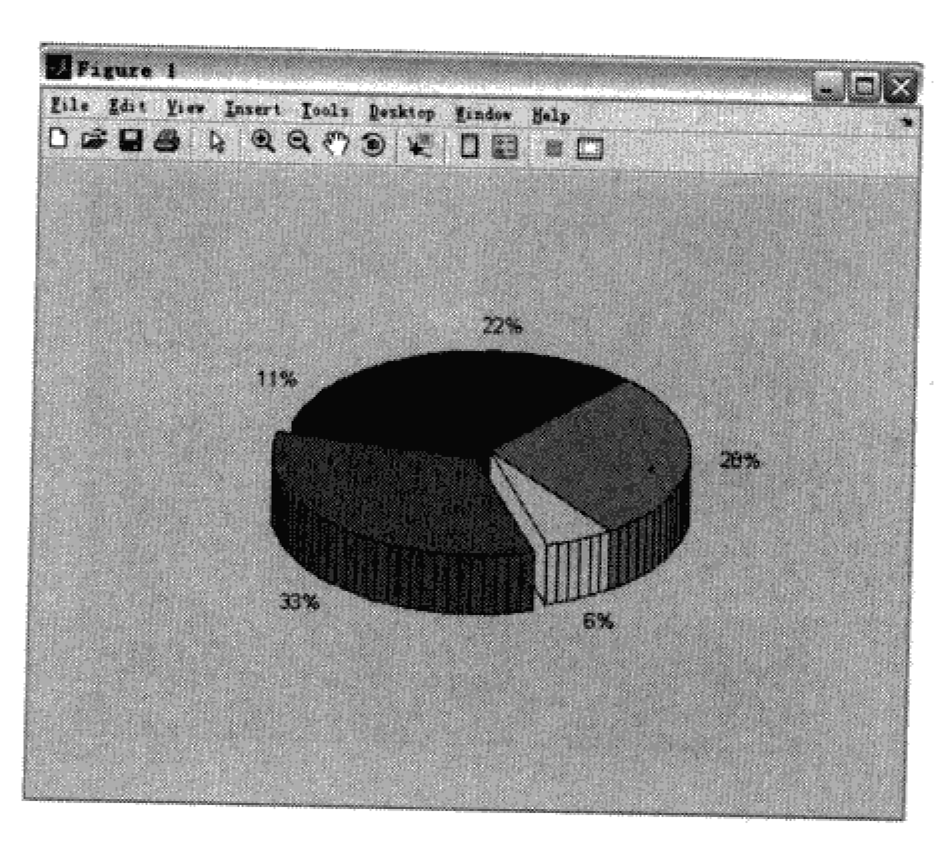


图 6.53 三维饼图

【实例讲解】 从上面的结果中可以看出，数组中的第二个数据从饼图中分离出去。

6.3.12　comet3 函数——三维彗星图绘制

【语法说明】

- comet3(z)：用向量 z 中的数据显示一个三维彗星。
- comet3(x,y,z)：显示一个彗星通过数据 x、y、z 确定的三维曲线。
- comet3(x,y,z,p)：指定彗星体的长度为 p*length（y）。

【功能介绍】 绘制三维空间中的彗星图。彗星图为一个三维的动画图像，彗星头（一个小圆圈）沿着数据指定的轨道前进，彗星体为跟在彗星头后面的一段痕迹，彗星轨道为整个函数所画的实曲线。

【实例 6.51】 绘制三维彗星图。

```
>>t = -20*pi:pi/50:20*pi;
>>comet3((cos(2*t).^2).*sin(t),(sin(2*t).^2).*cos(t),t);
```

得到的结果如图 6.54 所示。

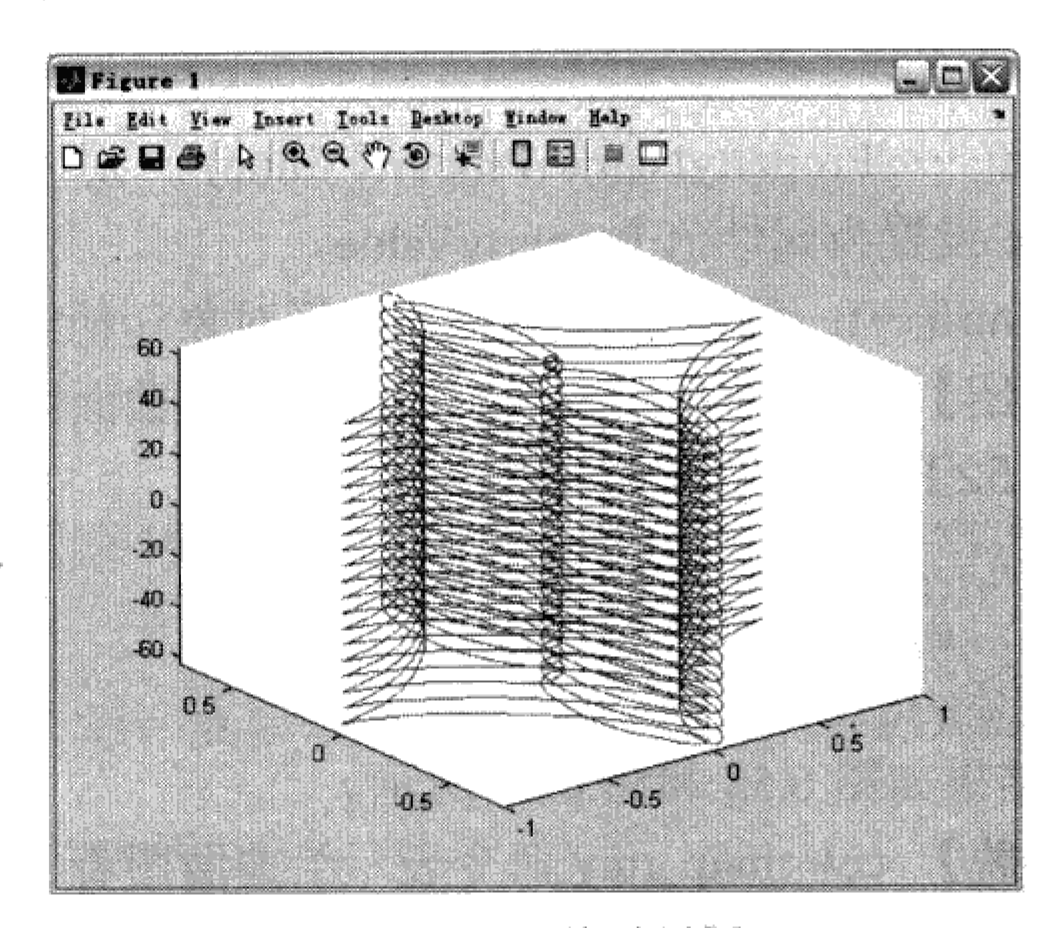


图 6.54　三维彗星图

【实例讲解】 用户可以在 MATLAB 中运行上面的函数，查看运行的结果。

6.3.13 surf 函数——阴影曲面图

【语法说明】

- surf(Z)：生成一个由矩阵 Z 确定的三维带阴影的曲面图，其中 [m，n] = size（Z），而 X = 1：n，Y=1：m。高度 Z 为定义在一个几何矩形区域内的单值函数，Z 同时指定曲面高度数据的颜色，所以颜色对于曲面高度是恰当的。
- surf(X,Y,Z)：数据 z 同时为曲面高度，也是颜色数据。X 和 Y 为定义 *x* 坐标轴和 *y* 坐标轴的曲面数据。若 X 与 Y 均为向量，length（X）=n，length（Y）=m，而[m,n]=size（Z），在这种情况下，空间曲面上的节点为（X(I),Y(j),Z(I,j)）。
- surf(X,Y,Z,C)：用指定的颜色 c 画出三维网格图。MATLAB 会自动对矩阵 c 中的数据进行线性变换，以获得当前色图中可用的颜色。
- surf(… ,'PropertyName',PropertyValue)：对指定的属性 PropertyName 设置为属性值 PropertyValue。
- h = surf(…)：返回一个 surface 图形对象句柄给变量 h。

【功能介绍】 在矩形区域内显示三维带阴影曲面图。

【实例 6.52】 绘制阴影曲面图。

```
>>[X,Y,Z] = peaks(30);
>>surf(X,Y,Z)
>>colormap gray
```

得到的结果如图 6.55 和图 6.56 所示。

【实例讲解】 colormap gray 命令表示给曲面图着了灰颜色。第三个命令得到了灰色的阴影图。

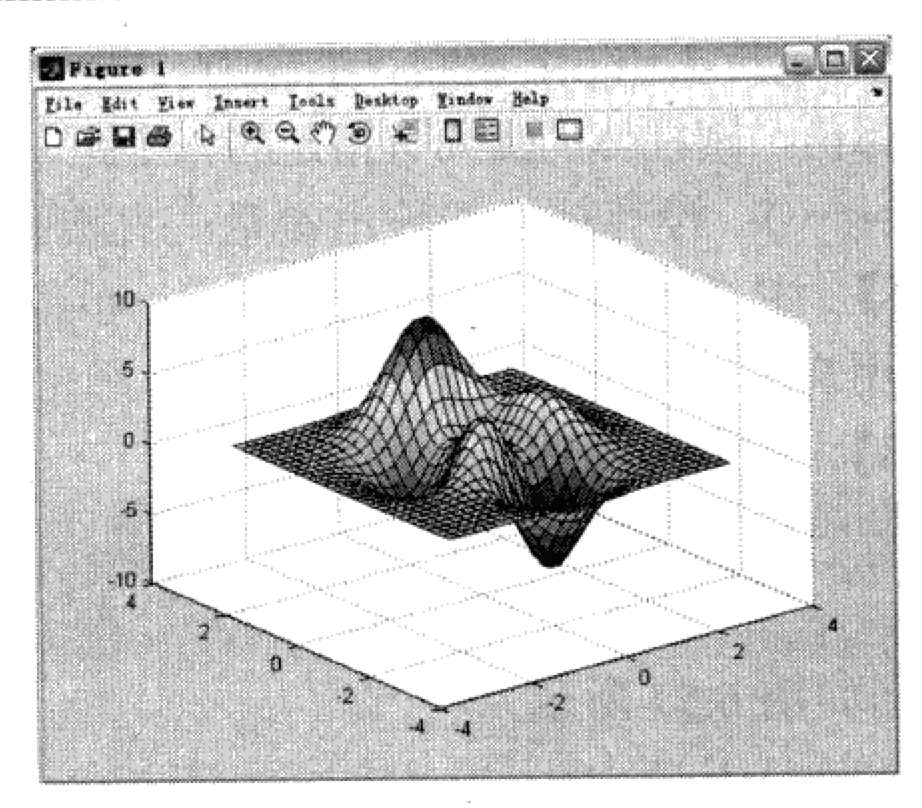


图 6.55　彩色阴影图

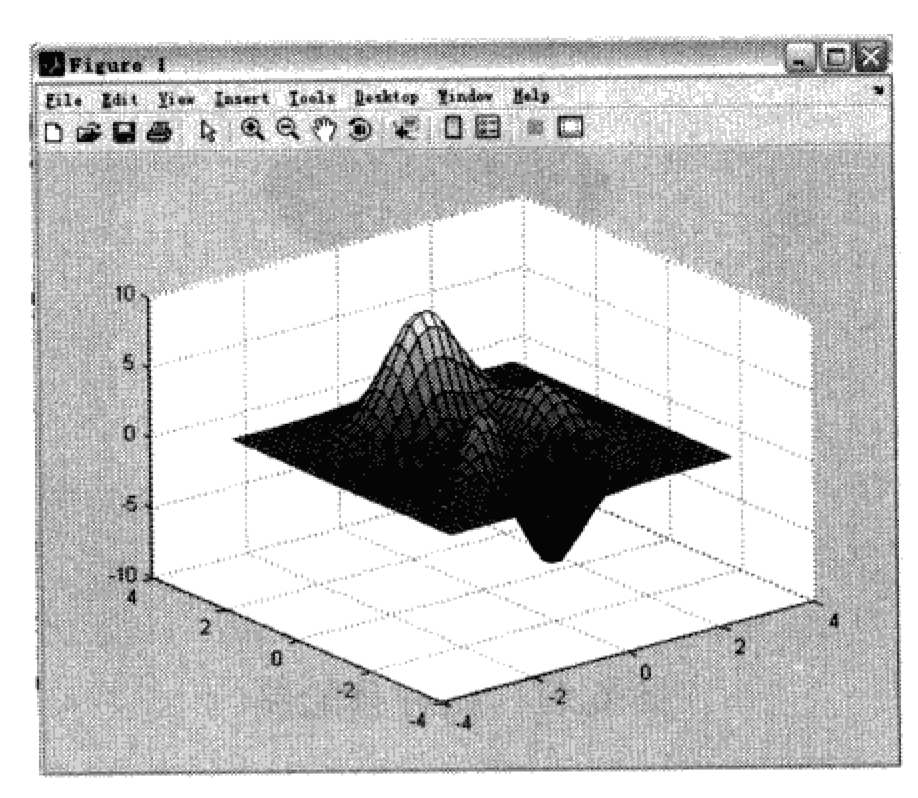


图 6.56　灰色阴影图

6.3.14　cylinder 函数——生成圆柱图形

【语法说明】

- [X,Y,Z] = cylinder：返回一半径为 1、高度为 1 的圆柱体的 *x*，*y*，*z* 轴的坐标值，圆柱体的圆周有 20 个距离相同的点。
- [X,Y,Z] = cylinder®：返回一半径为 r、高度为 1 的圆柱体的 *x*，*y*，*z* 轴的坐标值，圆柱体的圆周有 20 个距离相同的点。

- [X,Y,Z] = cylinder(r,n)：返回一半径为 r、高度为 1 的圆柱体的 *x*，*y*，*z* 轴的坐标值，圆柱体的圆周有指定的 *n* 个距离相同的点。
- cylinder(…)：没有任何的输出参量，直接画出圆柱体。

【功能介绍】 生成圆柱图形。该函数生成一单位圆柱体的 *x*，*y*，*z* 轴的坐标值。用户可以用函数 surf 或函数 mesh 画出圆柱形对象。

【实例 6.53】 绘制圆柱体。

```
>>t = 0:pi/10:2*pi;
>>[X,Y,Z] = cylinder(2+(cos(t)).^2);
>>surf(X,Y,Z);
```

绘制的图形如图 6.57 所示。

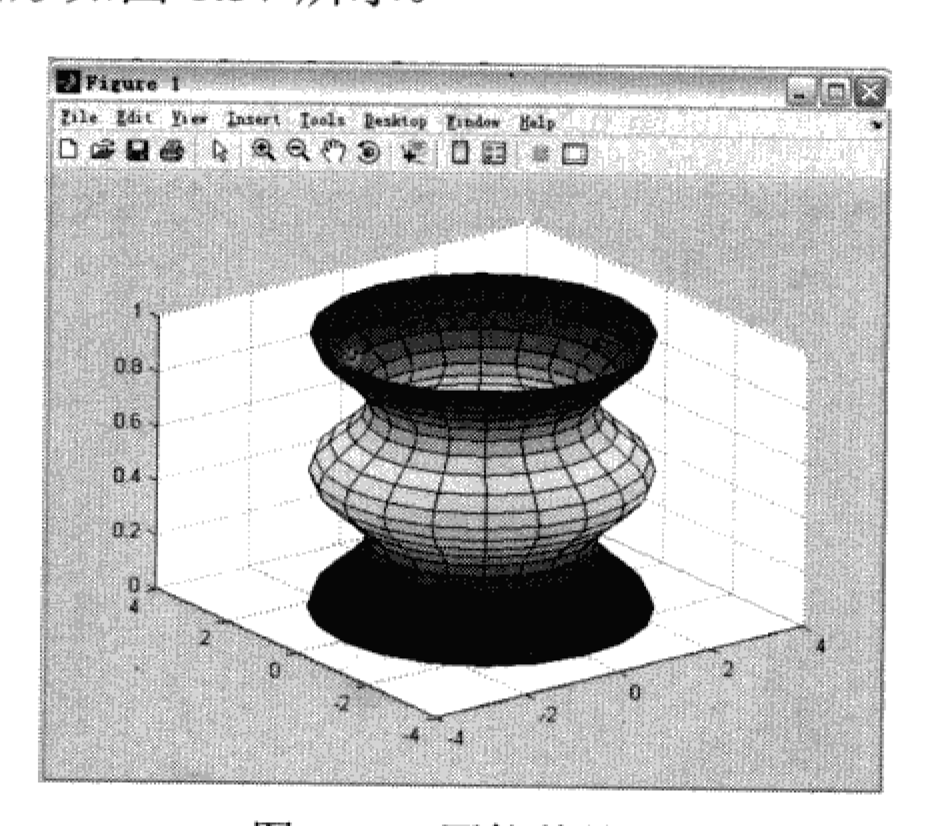


图 6.57 圆柱体绘制

【实例讲解】 X、Y、Z 的值通过 cylinder 得到。

【实例 6.54】 绘制一个旋转柱面图。

```
>> t=0:pi/12:3*pi;
>> r=abs(exp(-0.25*t).*sin(t));
>> [X,Y,Z]=cylinder(r,30);
mesh(X,Y,Z)
colormap([1 0 0])
```

绘制的图形如图 6.58 所示。

【实例讲解】 在上面的结果中，用户使用 colormap 函数为图形着色。

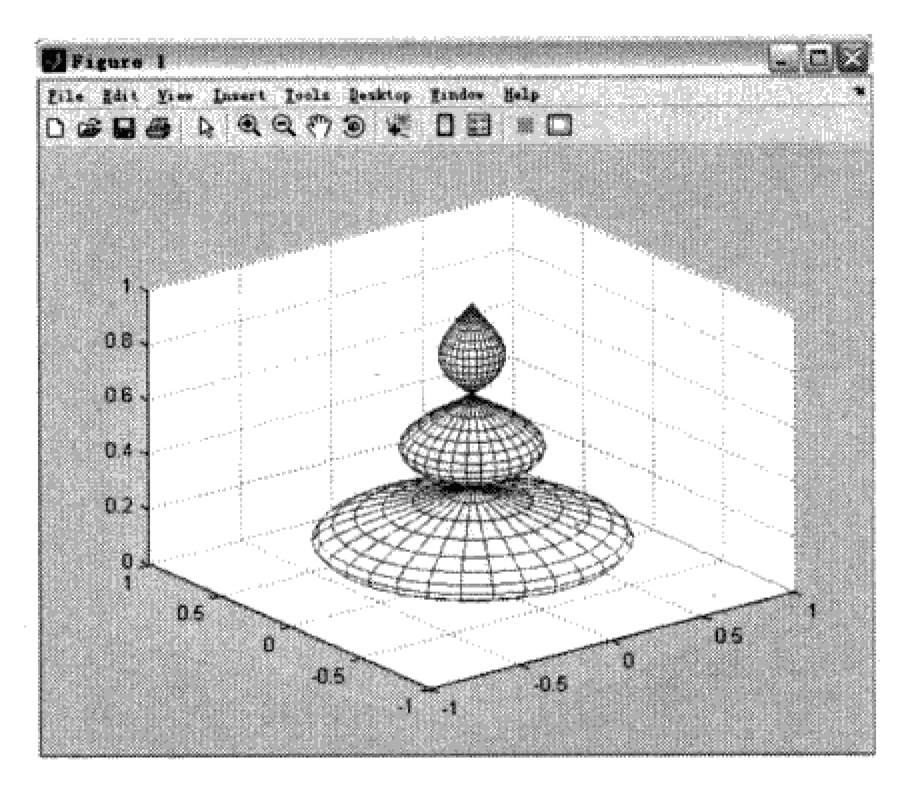


图 6.58　旋转柱面图

6.3.15　surfc 函数——绘制阴影图及等高线

【语法说明】

- surfc(Z)
- surfc(X,Y,Z)
- surfc(X,Y,Z,C)
- surfc(…,'PropertyName',PropertyValue)
- surfc(…)
- h = surfc(…)

上面各个使用形式的曲面效果与函数 surf 的相同，只不过是在曲面下面增加了曲面的等高线而已。

【功能介绍】 在矩形区域内显示三维带阴影曲面图，且在曲面下面画出等高线。

【实例 6.55】 绘制一个带阴影区的等高线。

```
>>[X,Y,Z] = peaks(30);
>>surfc(X,Y,Z)
>>colormap hsv
```

绘制的图形如图 6.59 所示。

【实例讲解】 最后一个命令加强了阴影的效果。

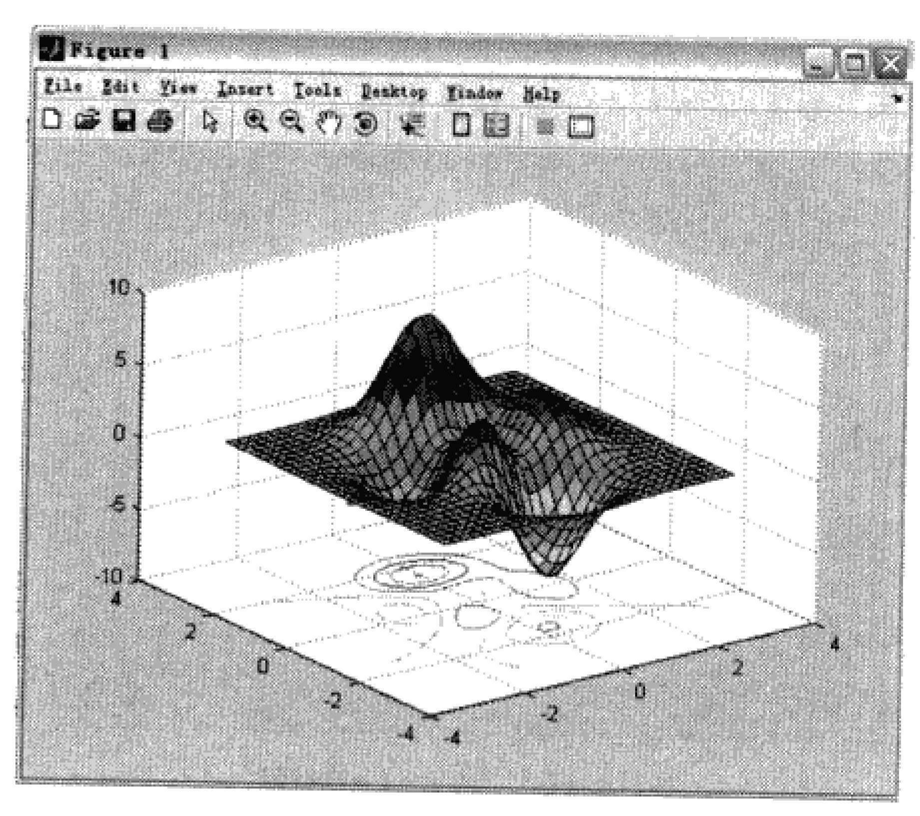


图 6.59　阴影与等高图

6.3.16　surfl 函数——带光照模式的曲面图

【语法说明】

- surfl(Z)：以向量 Z 的元素生成一个三维的带阴影的曲面，其中阴影模式中的光源的方位、光照系数为默认值（见下面）。
- surfl(X,Y,Z)：以矩阵 X、Y、Z 生成的一个三维的带阴影的曲面，其中阴影模式中的光源的方位、光照系数为默认值。
- surfl(…,'light')：用一个 matlab 光照对象（light object）生成一个带颜色、带光照的曲面，这与用默认光照模式产生的效果不同。
- surfl(…, 'cdata')：改变曲面颜色数据（color data），使曲面成为可反光的曲面。
- surfl(…,s)：指定光源与曲面之间的方位 s，其中 s 为一个二维向量[azimuth，elevation]，或者三维向量[sx，sy，sz]。默认光源方位为从当前视角开始，逆时针 45°。
- surfl(X,Y,Z,s,k)：指定反射常系数 k，其中 k 为一个定义环境光（ambient light）系数（0<=ka<=1）、漫反射（diffuse reflection）系数（0<=kb<=1）、镜面反射（specular reflection）系数（0<=ks<=1）

与镜面反射亮度（以相素为单位）等的四维向量[ka，kd，ks，shine]，默认值为 k=[0.55 0.6 0.4 10]。

- h = surfl(…)：返回一个曲面图形句柄向量 h。

【功能介绍】 画带光照模式的三维曲面图。该函数显示一个带阴影的曲面，结合了周围的，散射的和镜面反射的光照模式。想获得较平滑的颜色过渡，要使用有线性强度变化的色图（如 gray、copper、bone、pink 等）。参数 X、Y、Z 确定的点定义了参数曲面的“里面”和“外面”。

【实例 6.56】 光照模式的图形。

```
>>[X,Y] = meshgrid(-3:1/8:3);
>>Z = peaks(X,Y);
>>surfl(X,Y,Z);
>>shading interp
>>colormap(gray);
```

得到的图形如图 6.60 所示。

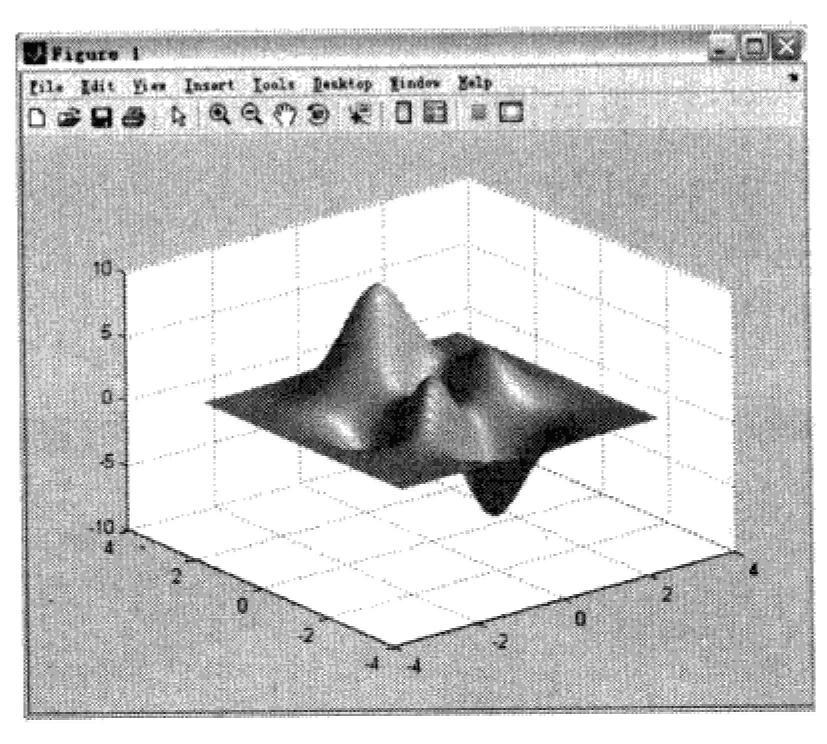


图 6.60 光照效果图

【实例讲解】 colormap(gray)对颜色进行了处理。

6.3.17 waterfall 函数——瀑布图

【语法说明】

- waterfall(X,Y,Z)：用所给参数 X、Y 与 Z 的数据画一“瀑

布”效果图。若 X 与 Y 都是向量，则 X 与 Z 的列相对应，Y 与 Z 的行相对应，即 length(X)=Z 的列数，length(Y)=Z 的行数。参数 X 与 Y 定义了 x 轴与 y 轴，Z 定义了 z 轴的高度，Z 同时确定了颜色，所以颜色能恰当地反映曲面的高度。若想研究数据的列，可以输入 waterfall(Z’)或 waterfall(X’,Y’,Z’)。

- waterfall(Z)：画出一瀑布图，其中默认地有：X=1:Z 的行数，Y=1:Z 的行数，且 Z 同时确定颜色，所以颜色能恰当地反映曲面高度。
- waterfall(…,C)：用比例化的颜色值从当前色图中获得颜色，参量 C 决定颜色的比例，为此，必须与 Z 同型。系统使用一线性变换，从当前色图中获得颜色。
- h = waterfall(…)：返回 patch 图形对象的句柄 h，可用于画出图形。

【功能介绍】 瀑布图。

【实例 6.57】 绘制瀑布图。

```
>>[X,Y,Z] = peaks(20);
>>waterfall(X,Y,Z)
```

得到的结果如图 6.61 所示。

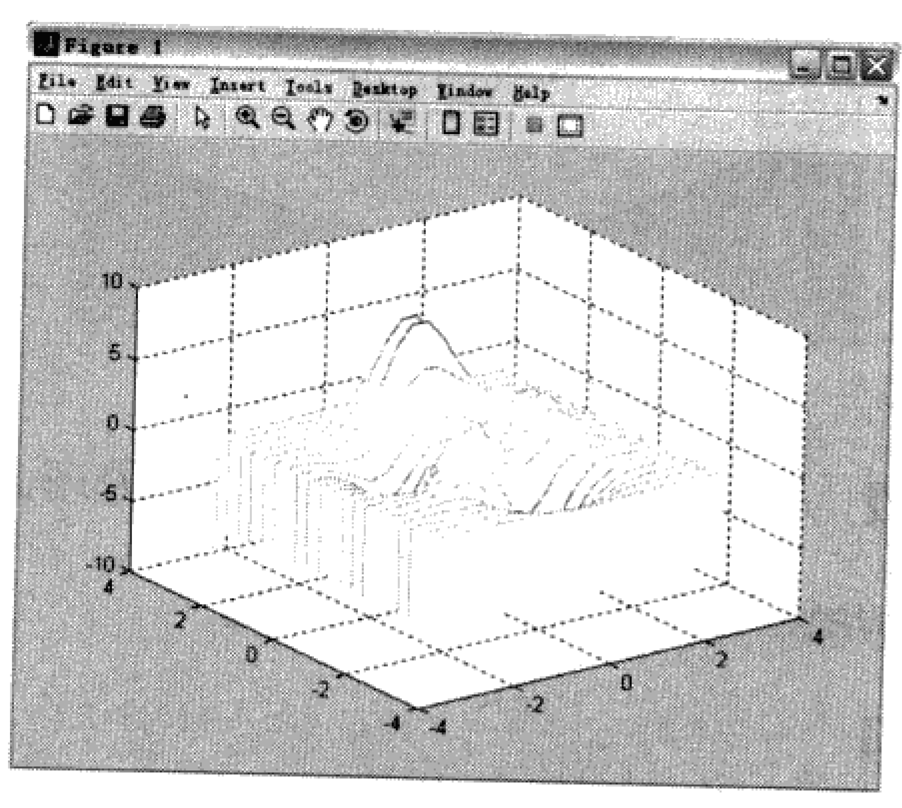


图 6.61 瀑布图

【实例讲解】 waterfall(X,Y,Z)函数得到的瀑布图如图 6.61 所示。

6.4 图形图像处理与动画制作

在一般情况下，MATLAB 能够绘制比较好的图形图像，不过在一些特殊的场合，或者一些特殊的要求的情况下，需要对图像进行处理。而且，利用 MATLAB 的一些函数可以很容易地实现动画制作功能。

6.4.1 view 函数——视点处理

【语法说明】

- view(az,el)、view([az,el])：给三维空间图形设置观察点的方位角。方位角 az 与仰角 el 为这两个旋转角度：做一通过视点与 z 轴的平面，与 xy 平面有一交线，该交线与 y 轴的反方向的、按逆时针方向（从 z 轴的方向观察）计算的、单位为度的夹角，就是观察点的方位角 az。
- view([x,y,z])：在笛卡儿坐标系中于点（x,y,z）设置视点。注意：输入参量只能是方括号的向量形式，而非数学中的点的形式。
- view(2)：设置默认的二维形式视点。其中 az=0，el=90，即从 z 轴上方观看。
- view(3)：设置默认的三维形式视点。其中 az=-37.5，el=30。
- view(T)：根据转换矩阵 T 设置视点。其中 T 为 4×4 阶的矩阵，如同用函数 viewmtx 生成的透视转换矩阵一样。
- [az,el] = view：返回当前的方位角 az 与仰角 el。
- T = view：返回当前的 4×4 阶的转换矩阵 T。

【功能介绍】 指定立体图形的观察点。用户可以用方位角（azimuth）和仰角（elevation）一起，或者用空间中的一点来确定观察点的位置。

【实例 6.58】 生成一个多峰函数，并对其进行视点处理。

```
>>peaks;
>>az = 30;el = 90;
>>view(az, el)
```

处理结果如图 6.62 所示。

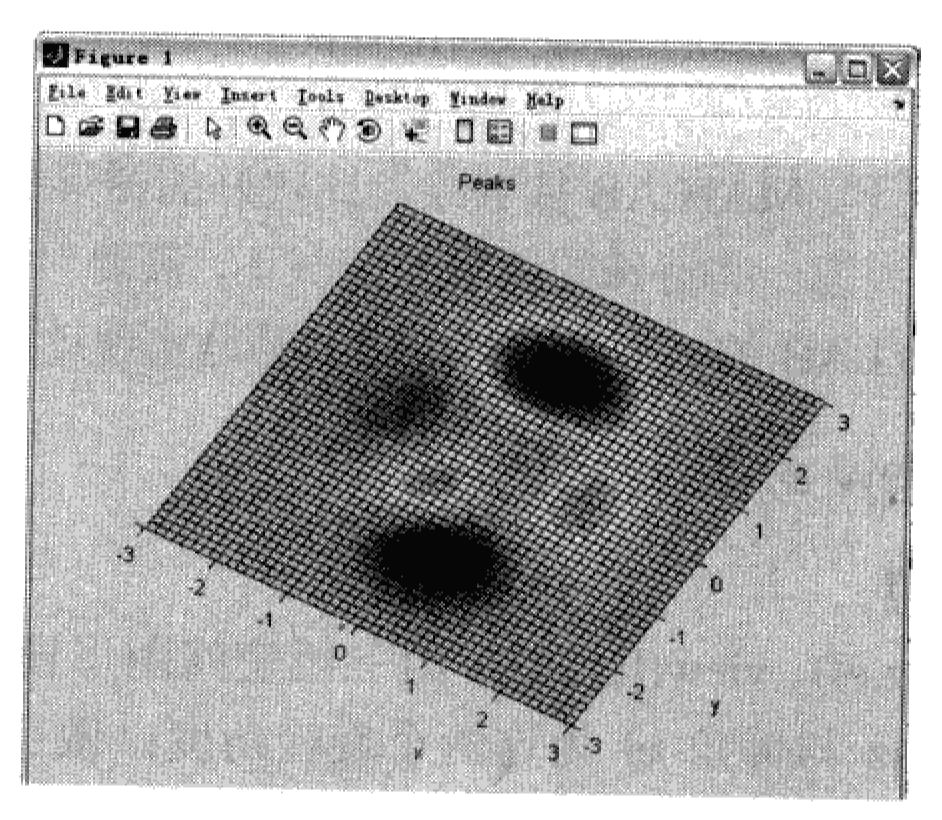


图 6.62　视点处理

【实例讲解】 以上这个处理取方位角为 30°，仰角为 90°。

6.4.2 colormap 函数——获取当前色图

【语法说明】 colormap(map)：通过矩阵 map 设置色图。若矩阵 map 中的元素不在[0 1]区间之内，则返回一个错误。MATLAB 支持的色图如下。

- Cool：青蓝和洋红的色度。
- Bone：带一点蓝色的灰度。
- Flag：交替为红色、白色、蓝色和黑色。
- Jet：Hsv 的一种变形（以兰色开始和结束）。
- Copper：线性铜色度。
- Hsv：色彩饱和值（以红色开始和结束）。
- Hot：从黑色到黄色到白色。
- Gray：线性灰度。

- Pink：粉红的彩色度。
- Prim：三棱镜。交替为红色、橘黄色、黄色、绿色和天蓝色。
- Lines：线性色图。
- White：全白色图。
- Colorcube：增强立方色图。
- Autumn：红色黄色阴影色图。
- Spring：洋红黄色阴影色图。
- Summer：绿色黄色阴影色图。
- Winter：兰色绿色阴影色图。

【功能介绍】 设置或获取当前色图。色图为一个 m×3 的、元素在 0～1 之间的实数的矩阵，每一行为定义一个颜色的 RGB 向量。色图矩阵的第 *k* 行定义了第 *k* 个颜色，其中 map(k,:)=[r(k) g(k) b(k)]指定了组成该颜色中红色、绿色、兰色的强度。

【实例 6.59】 制作一个黄绿色的球体。

```
>>[x,y,z]=sphere(20);
>>colormap(Summer);
>> surf(x,y,z);
```

生成的球体如图 6.63 所示。

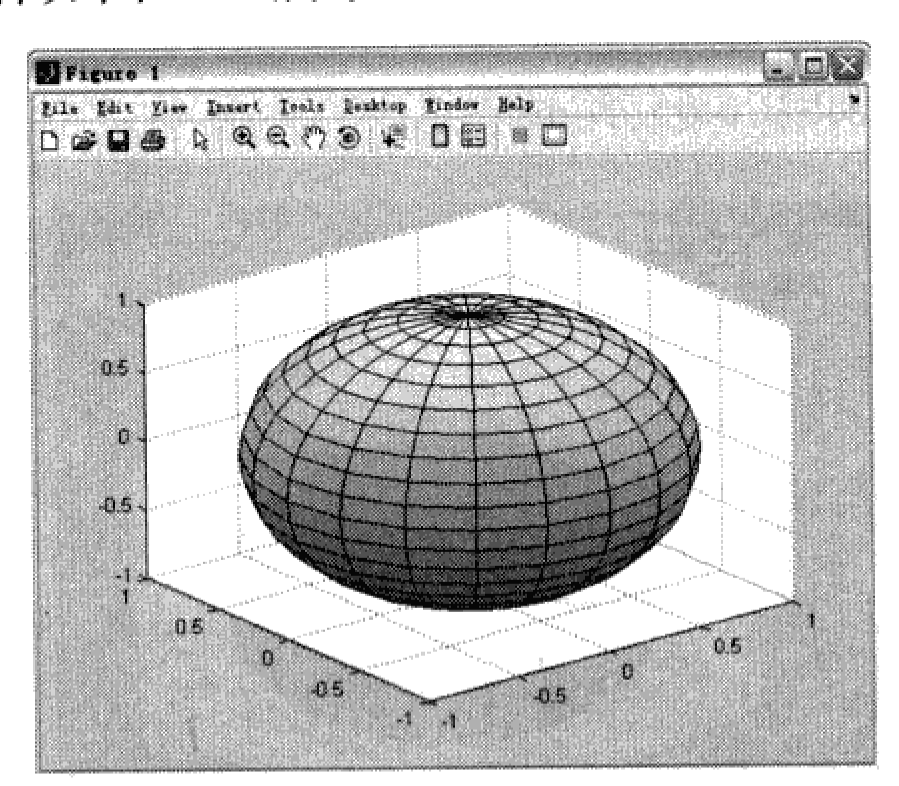


图 6.63 对球体设置色图

【实例讲解】 colormap(Summer)进行了上色处理。

【实例 6.60】 生成 64 种颜色的 hsv 色图的球体。

```
>> [x,y,z]=sphere(20);
>> colormap(cool(64));
>> surf(x,y,z);
```

得到的图形如图 6.64 所示。

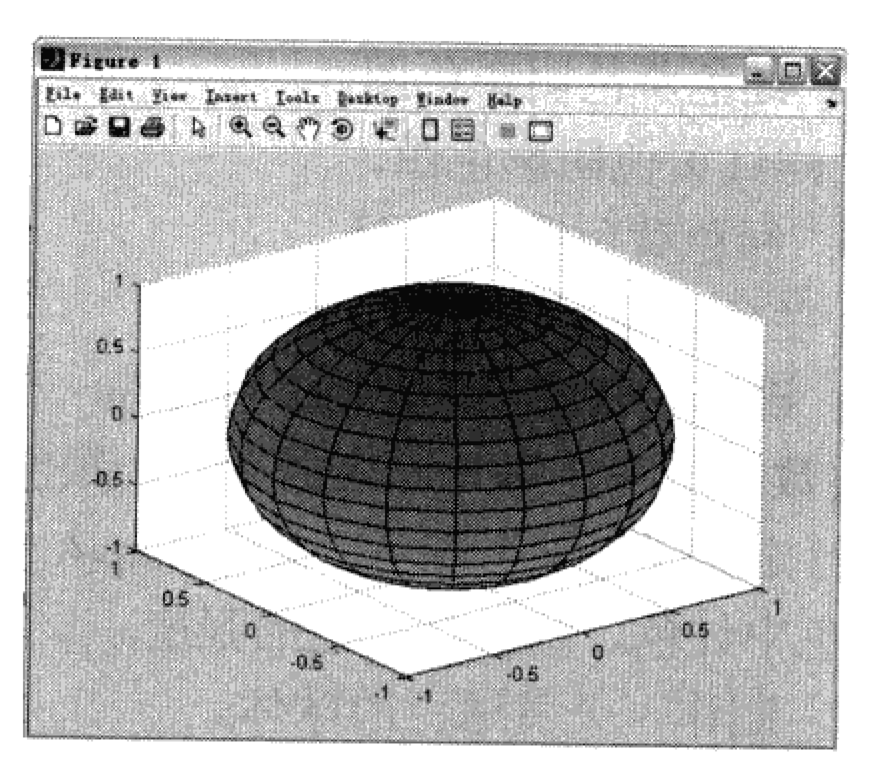


图 6.64　64 位色图

```
>> [x,y,z]=sphere(20);
>> colormap(cool(4));
>> surf(x,y,z);
```

得到的图形如图 6.65 所示

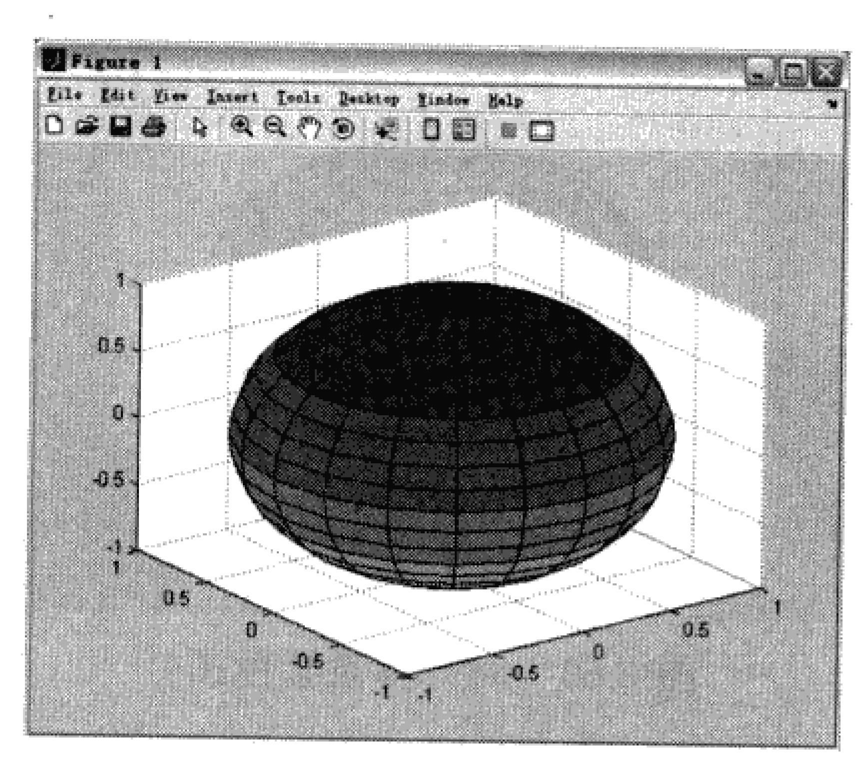


图 6.65　4 位色图

【实例讲解】 通过上面两个图形可以看出两者的区别，64 种颜色色图的图形过渡性更好。

6.4.3　brighten 函数——色图控制函数

【语法说明】

- brighten(beta)：增亮或变暗当前的色图。若 0<beta<1，则增亮色图；若-1<beta<0，则变暗色图。改变的色图将代替原来的色图，但本质上是相同的颜色只是颜色的亮度不同。
- brighten(h,beta)：对指定的句柄对象 h 中的子对象进行操作。
- newmap = brighten(beta)：该函数没有改变当前图形的亮度，而是返回变化后的色图给 newmap。
- newmap = brighten(cmap,beta)：该函数没有改变指定色图 cmap 的亮度，而是返回变化后的色图给 newmap。

【语法说明】 增亮或变暗色图。

6.4.4　colorbar 函数——显示颜色条

【语法说明】

- colorbar：更新最近生成的颜色条，或若当前坐标轴没有一颜色条，则在右边显示一垂直的颜色条。
- colorbar('vert')：增加一垂直的颜色条到当前的坐标轴。
- colorbar('horiz')：增加一水平的颜色条到当前的坐标轴。
- colorbar(h)：用坐标轴 h 来生成一颜色条。若坐标轴的宽度大于高度，则颜色条是水平放置的。
- h = colorbar(…)：返回一颜色条句柄 h，该句柄是一坐标轴对象。
- colorbar(…,'peer',axes_handle)：生成一与坐标轴 axes-handle 有关的颜色条，代替当前的坐标轴。

【功能介绍】 显示能指定颜色刻度的颜色条，且调整当前坐标

轴，以适应当前的颜色条。

【实例 6.61】 在球体图形中显示颜色刻度条。

```
>> [x,y,z]=sphere(20);
>> surf(x,y,z);
>> colorbar('vert')    %加垂直颜色条
>> colorbar('horiz')   %加水平颜色条
```

得到的结果如图 6.66 所示。

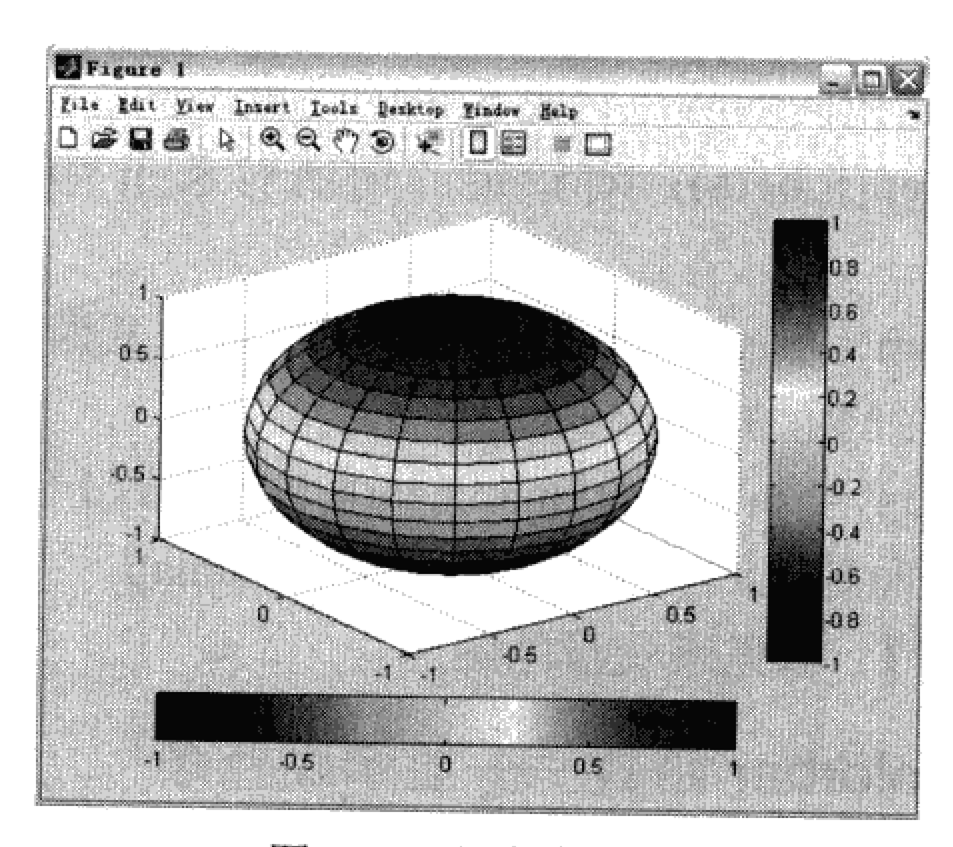


图 6.66　添加刻度条

【实例讲解】 在球体图形中添加了垂直刻度条和水平刻度条。

6.4.5　contrast 函数——提高灰色对比度

【语法说明】

- cmap = contrast(X)：返回一灰度色图，该色图与当前色图有相同的维数。参量 cmap 为生成的灰度色图。
- cmap = contrast(X,m)：返回维数为 m×3 的灰度色图 cmap。

【功能介绍】 提高灰度色图的对比度。该函数可以增强图像的对比度。

【实例 6.62】 sphere 用来绘制球形图，surf 函数用来进行平面处理和上色，colorbar('vert')中的参数指明绘制哪种类型的颜色条，

'vert'用来加垂直的，'horiz'用来加水平的颜色条。

```
>>load clown;
>>cmap = contrast(X);
>>image(X);
>>colormap(cmap);
```

6.4.6　rgbplot 函数——画出色图

【语法说明】 rgbplot(cmap)：画出维数为 m×3 的色图矩阵 cmap 的每一列，矩阵的第一列为红色强度，第二列为绿色强度，第三列为蓝色强度。

【功能介绍】 画出色图。

【实例 6.63】 画出载入图形的色图矩阵。

```
>> load clown;                  %加载图形
>> cmap = contrast(X);
>> rgbplot(cmap)
```

得到的色图矩阵如图 6.67 所示。

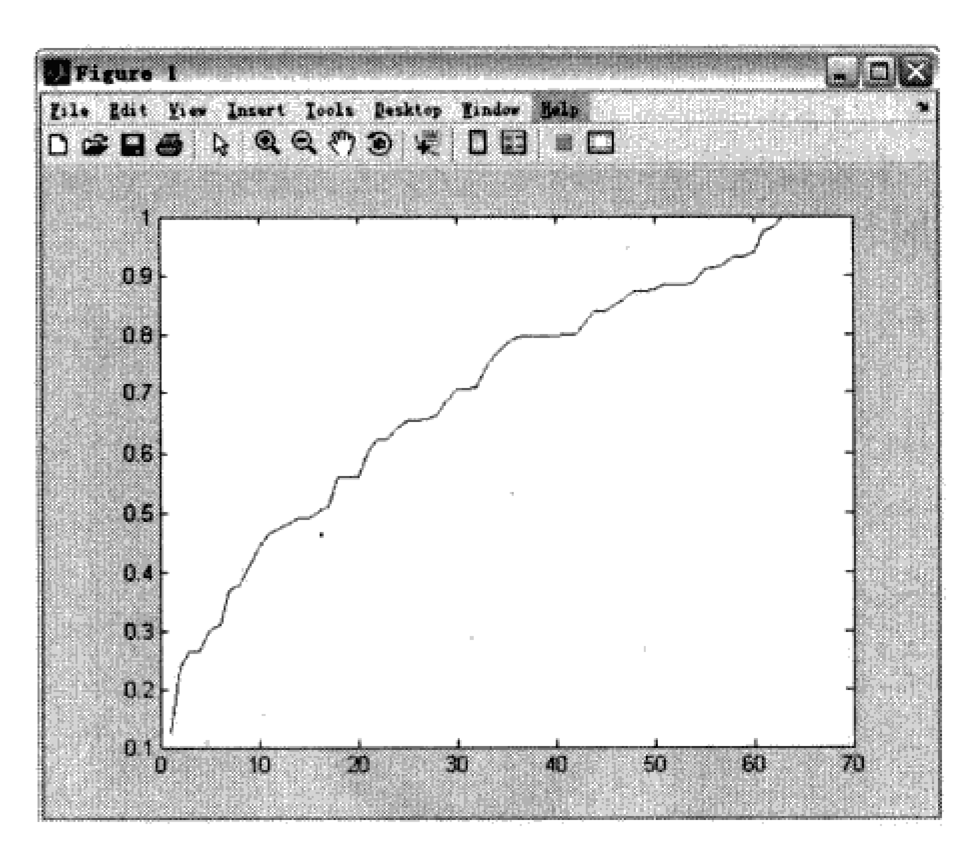


图 6.67　色图矩阵

【实例讲解】 clown 是 MATLAB 自带的数据矩阵。在使用之前，用户需要首先加载。

6.4.7 shading 函数——设置颜色色调

【语法说明】

- shading flat：使网格图上的每一线段与每一小面有一相同颜色，该颜色由线段末端的端点颜色确定；或由小面的、有小型的下标或索引的 4 个角的颜色确定。
- shading faceted：带重叠的黑色网格线的平面色调模式。这是默认的色调模式。
- shading interp：在每一线段与曲面上显示不同的颜色，该颜色为通过在每一线段两边的，或者为不同小曲面之间的色图的索引或真颜色进行内插值得到的颜色。

【功能介绍】 设置颜色色调属性。该函数控制曲面与补片等的图形对象的颜色色调。函数 shading 设置恰当的属性值，取决于曲面或补片对象是表现网格图或实曲面。

【实例 6.64】 没有经过着色处理和经过平滑处理的比较。

```
>>sphere(20)
>>axis square
>>shading flat
>>title('平滑处理')
```

得到的图形如图 6.68 和图 6.69 所示。

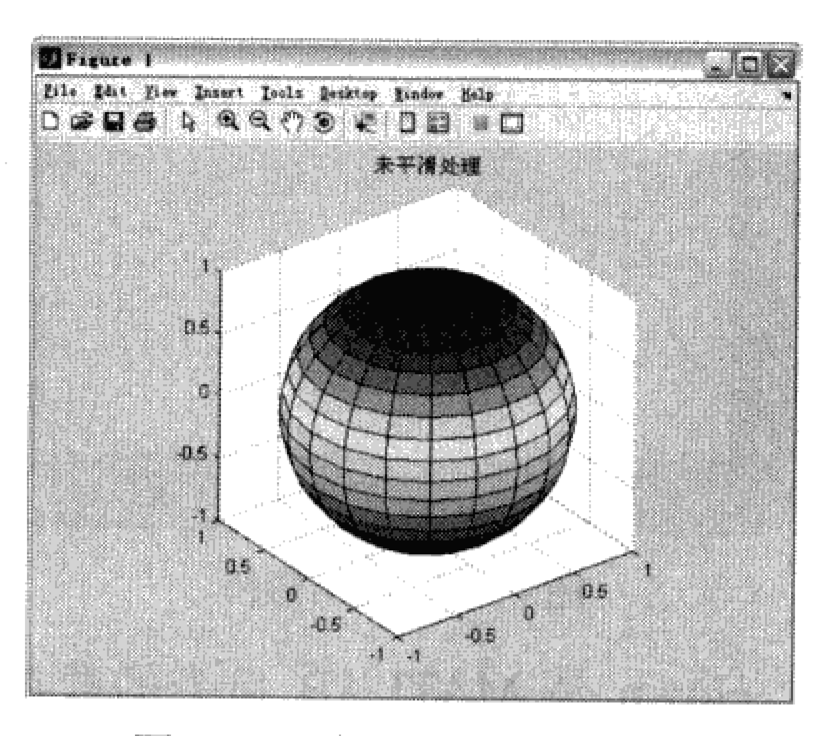


图 6.68 未进行平滑处理图

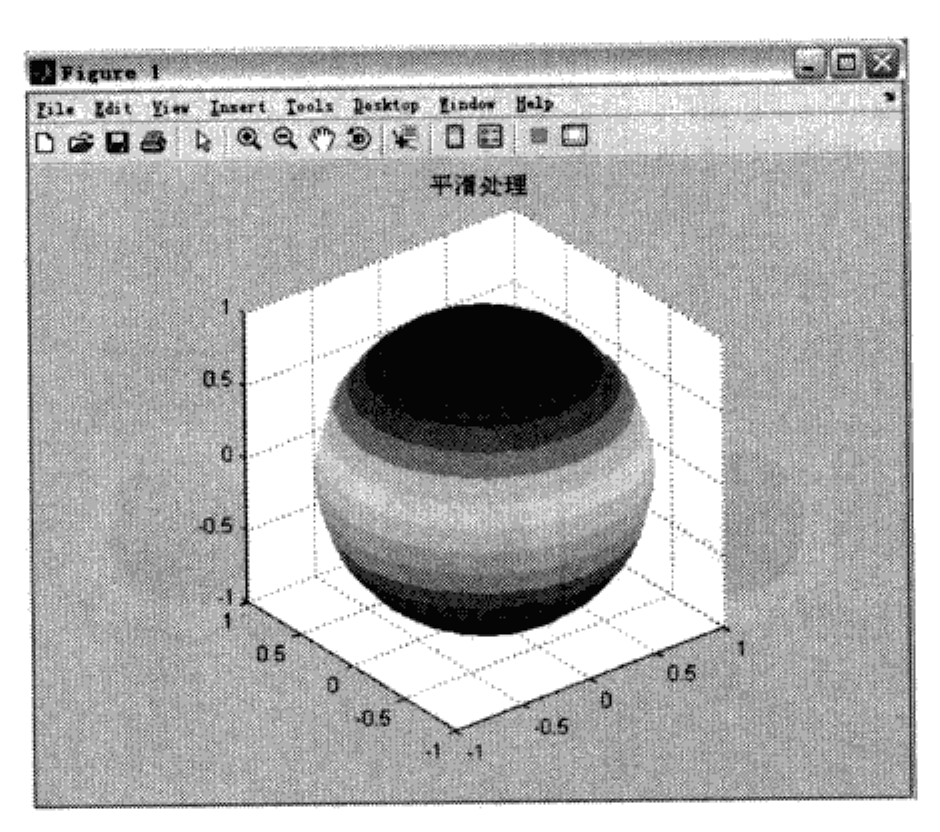


图 6.69　设置平滑色调

【实例讲解】 从上面两个图形的比较可以看出，经过平滑色调后的图形更加美观。

【实例 6.65】 3 种不同着色效果图的演示。

```
[x,y,z]=sphere(20);
colormap(copper);
subplot(1,3,1);
surf(x,y,z);
axis equal
subplot(1,3,2);
surf(x,y,z);
shading flat;
axis equal
subplot(1,3,3);
surf(x,y,z);
shading interp;
axis equal
```

得到的结果如图 6.70 所示。

【实例讲解】 从理论上讲，着色不同是由于图形的计算密度不同。

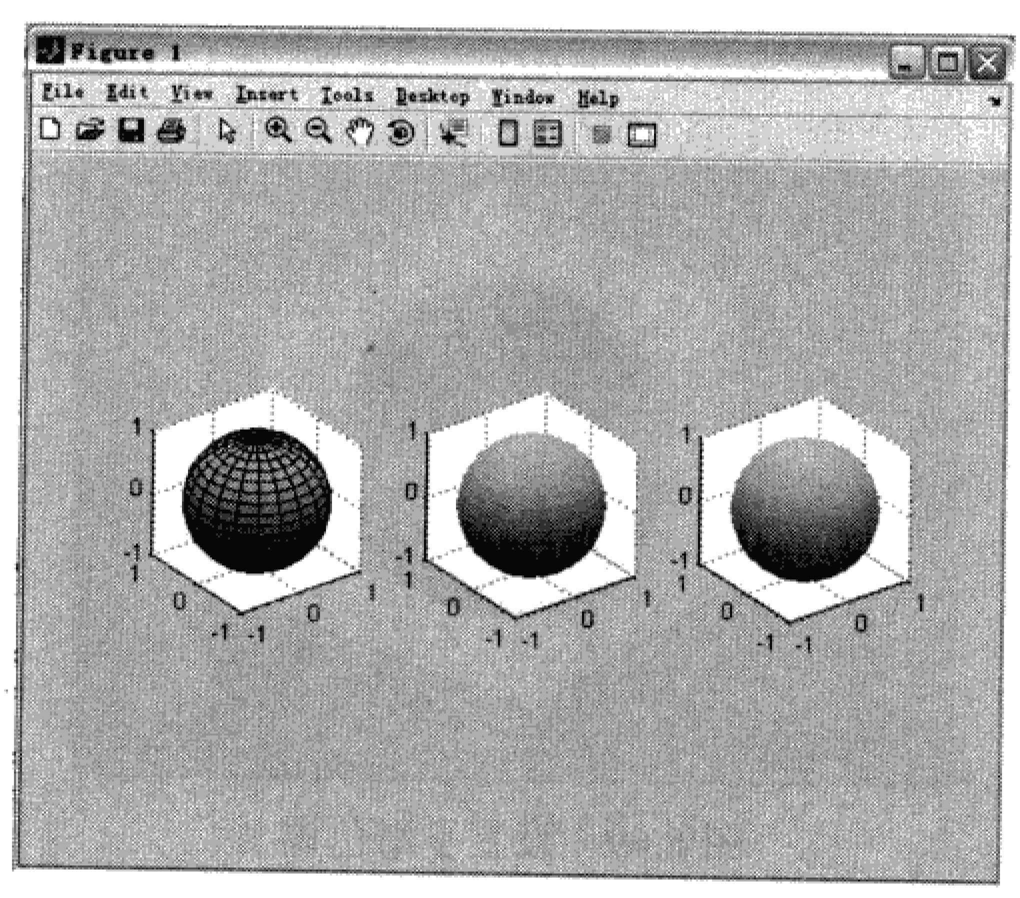


图 6.70　三种着色效果图比较

6.4.8　hidden 函数——隐含线条的显示

【语法说明】

- hidden on：对当前图形打开隐含线条的显示状态，使网格图后面的线条被前面的线条遮住。设置曲面图形对象的属性 FaceColor 为坐标轴背景颜色。这是系统的默认操作。
- hidden off：对当前图形关闭隐含线条的显示。
- hidden：在两种状态 on 与 off 之间切换。

【功能介绍】 在一网格图中显示隐含线条。隐含线条的显示，实际上是显示那些从观察角度观看没有被其他物体遮住的线条。

【实例 6.66】 显示等高线中的水平面网格。

```
>>mesh(peaks)
>>hidden off
```

得到的图形如图 6.71 所示。

【实例讲解】 hidden off 关闭隐含线条的显示。

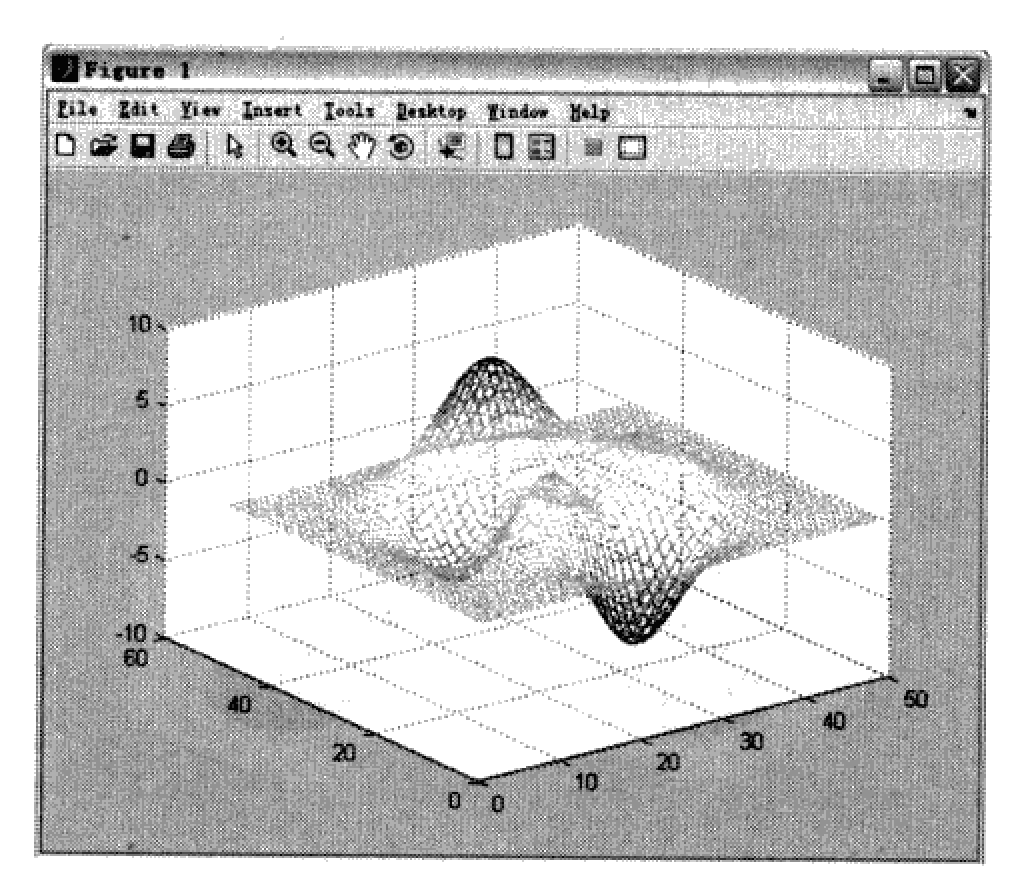


图 6.71 隐含线条的显示

6.4.9 light 函数——光照处理

【语法说明】 light('Color',选项 1,'Style',选项 2,'Position',选项 3)。

【功能介绍】 对所画的图形进行光照处理。

【实例 6.67】 不同光照处理的比较。

```
[x,y,z]=sphere(20);
subplot(1,2,1);
surf(x,y,z);
axis equal;
light('Posi',[0,1,1]);
shading interp;
hold on;
plot3(0,1,1,'p');
text(0,1,1,' light');
subplot(1,2,2);
surf(x,y,z);
axis equal;
light('Posi',[1,0,1]);
shading interp;
hold on;
plot3(1,0,1,'p');
text(1,0,1,' light');
```

得到的图形如图 6.72 所示。

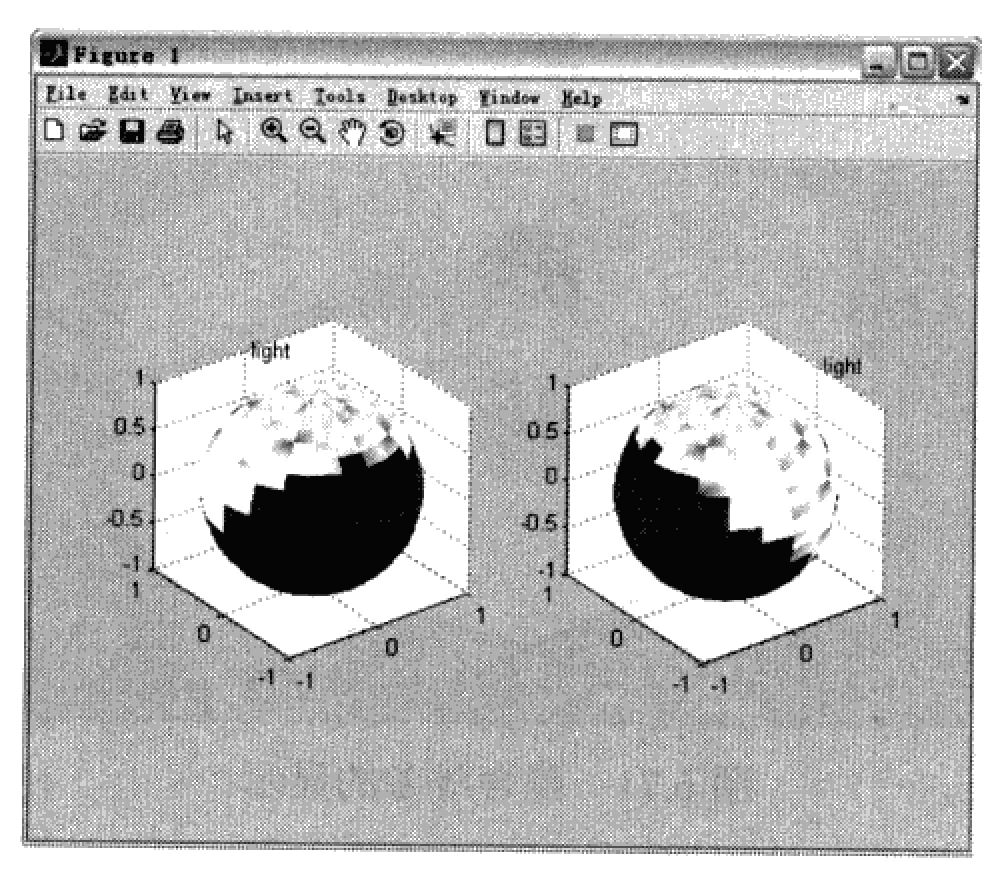


图 6.72　不同光照处理的比较

【实例讲解】 在这个例子中出现的函数在上面都已经讲解过，不再赘述。

6.4.10　图像的压缩和解压

【实例 6.68】 通过 M 文件建立用户自定义界面，实现图像的压缩和解压功能。

```
>> h0=figure('toolbar','none',...
    'position',[198 56 350 468],...
    'name','函数变换');
h1=axes('parent',h0,...
    'position',[0.25 0.45 0.5 0.5],...
    'visible','off');
I=imread('cameraman.tif');
imshow(I)
b1=uicontrol('parent',h0,...
    'units','points',...
    'tag','b1',...
    'backgroundcolor',[0.75 0.75 0.75],...
    'style','pushbutton',...
```

```
    'string','图像压缩',...
    'position',[30 100 50 20],...
    'callback',[...
        'cla,',...
        'I=imread(''cameraman.tif'');,',...
        'I2=im2double(I);,',...
        'imshow(I2)']);
b2=uicontrol('parent',h0,...
    'units','points',...
    'tag','b2',...
    'backgroundcolor',[0.75 0.75 0.75],...
    'style','pushbutton',...
    'string','图像解压',...
    'position',[100 100 50 20],...
    'callback',[...
        'cla,',...
        'I=imread(''cameraman.tif'');,',...
        'I=im2double(I);,',...
        'T=dctmtx(8);,',...
        'B=blkproc(I,[8 8],''P1*x*P2'',T,T'');,',...
        'mask=[1 1 1 1 0 0 0 0;,',...
              '1 1 1 0 0 0 0 0;,',...
              '1 1 0 0 0 0 0 0;,',...
              '1 0 0 0 0 0 0 0;,',...
              '0 0 0 0 0 0 0 0;,',...
              '0 0 0 0 0 0 0 0;,',...
              '0 0 0 0 0 0 0 0;,',...
              '0 0 0 0 0 0 0 0];,',...
        'B2=blkproc(B,[8 8],''P1.*x'',mask);,',...
        'I2=blkproc(B2,[8 8],''P1*x*P2'',T'',T);,',...
        'imshow(I2)']);
b3=uicontrol('parent',h0,...
    'units','points',...
    'tag','b3',...
    'backgroundcolor',[0.75 0.75 0.75],...
    'style','pushbutton',...
    'string','线条解析',...
    'position',[170 100 50 20],...
    'callback',[...
        'cla,',...
        'I=imread(''cameraman.tif'');,',...
```

```
            'BW=edge(I);,',...
            'imshow(BW)']);
    b4=uicontrol('parent',h0,...
        'units','points',...
        'tag','b4',...
        'backgroundcolor',[0.75 0.75 0.75],...
        'style','pushbutton',...
        'string','关闭',...
        'fontsize',15,...
        'position',[80 50 80 30],...
        'callback','close');
```

得到的图形如图 6.73 所示。



图 6.73　图像的压缩和解压

【实例讲解】 用户可以自行检测上面的程序结果。

6.4.11　图形的裁剪处理

【实例 6.69】 绘制三维曲面图，并进行插值着色处理，裁掉图中 x 和 y 都小于 0 的部分。

```
    [x,y]=meshgrid(-5:0.1:5);
    z=cos(x).*cos(y).*exp(-sqrt(x.^2+y.^2)/4);
```

```
surf(x,y,z);shading interp;
pause
i=find(x<=0&y<=0);
z1=z;z1(i)=NaN;
surf(x,y,z1);shading interp;
```

最后的图形结果如图 6.74 所示。

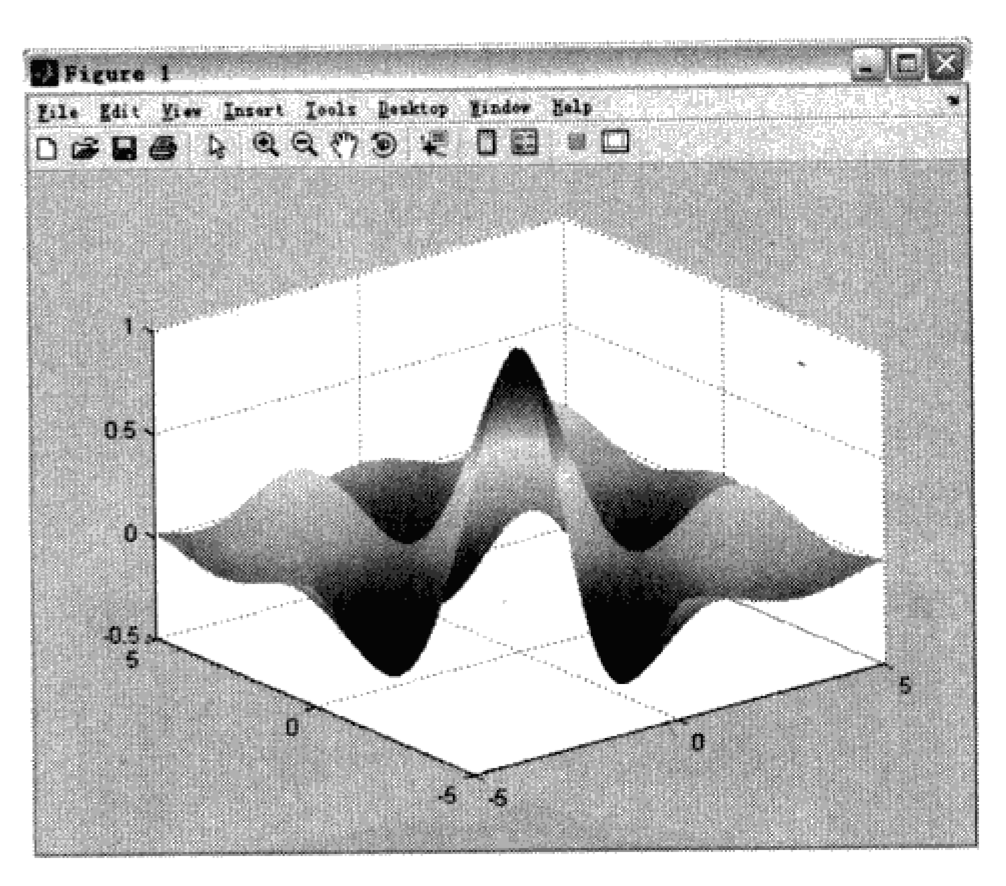


图 6.74 剪裁效果

【实例讲解】 pause 为暂停命令。为了显示效果，第一个曲面绘制完成后暂停，而后显示裁剪后的曲面。

6.4.12 hidden 函数——图像的消隐处理

【语法说明】

- hidden off：对图像取消做消隐处理。
- hidden on：对图像做消隐处理。

【功能介绍】 对图像做消隐处理。

【实例 6.70】 比较消隐前后的两幅图形。

```
>> z=peaks(50);

subplot(2,1,1);
```

```
mesh(z);
title('消隐前的网图')
hidden off
subplot(2,1,2)
mesh(z);
title('消隐后的网图')
hidden on
colormap([0 0 1])
```

最后的图形结果如图 6.75 所示。

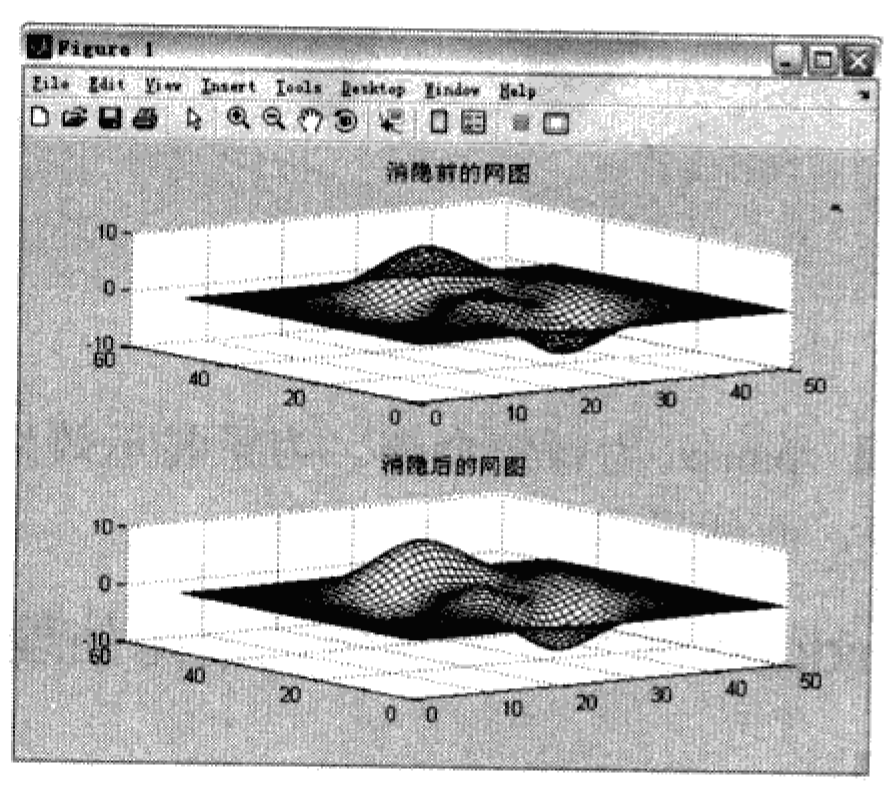


图 6.75　图形的消隐处理

【实例讲解】 上面的为消隐前的网格图，下面的一幅图为消隐后的网格图。

6.4.13　imread 和 imwrite 函数——读入读出图像文件

【语法说明】

- [x,cmap]=imread('filename')：将图像文件读入 MATLAB 工作

空间，MATLAB 支持多种图像文件格式，如.bmp、.jpg、.jpeg、.tif 等。

- [x,cmap]=imwrite('filename')：将图像数据和色图数据一起写入一定格式的图像文件，MATLAB 支持多种图像文件格式，如.bmp、.jpg、.jpeg、.tif 等。

【功能介绍】 读入或读出图像文件。

【实例 6.71】 读入风景图片文件“134.jpg”，并显示该图片。

```
>> [x,cmap]=imread('134.jpg');
>> image(x);
>>[x,cmap]=imwrite('cream.bmp');
```

得到的结果如图 6.76 所示。

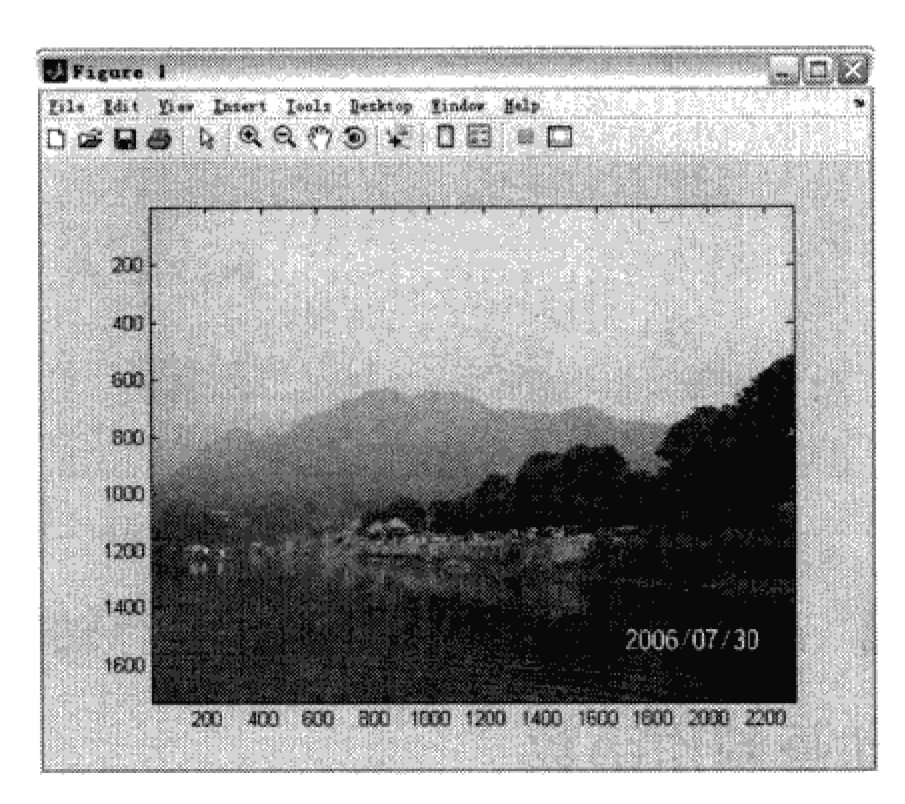


图 6.76　读入图像文件

【实例讲解】 读入的图片将该文件写入“cream.bmp”文件中。

6.4.14　image 和 imagesc 函数——显示图像文件

【语法说明】

- image(x)：x 为图像的数据阵。
- imagesc(x)：x 为图像的数据阵。

【功能介绍】 这两个函数用于图像显示。为了保证图像的显示效果，一般还应使用 colormap 函数设置图像色图。

【实例 6.72】 有一图像文件 134.jpg，在图形窗口取消坐标轴后显示该图像。

```
[x,cmap]=imread('boy.jpg');  %读取图像的数据阵和色图阵
image(x);
colormap(cmap);
axis image off    %保持宽高比并取消坐标轴
```

得到的图像如图 6.77 所示。

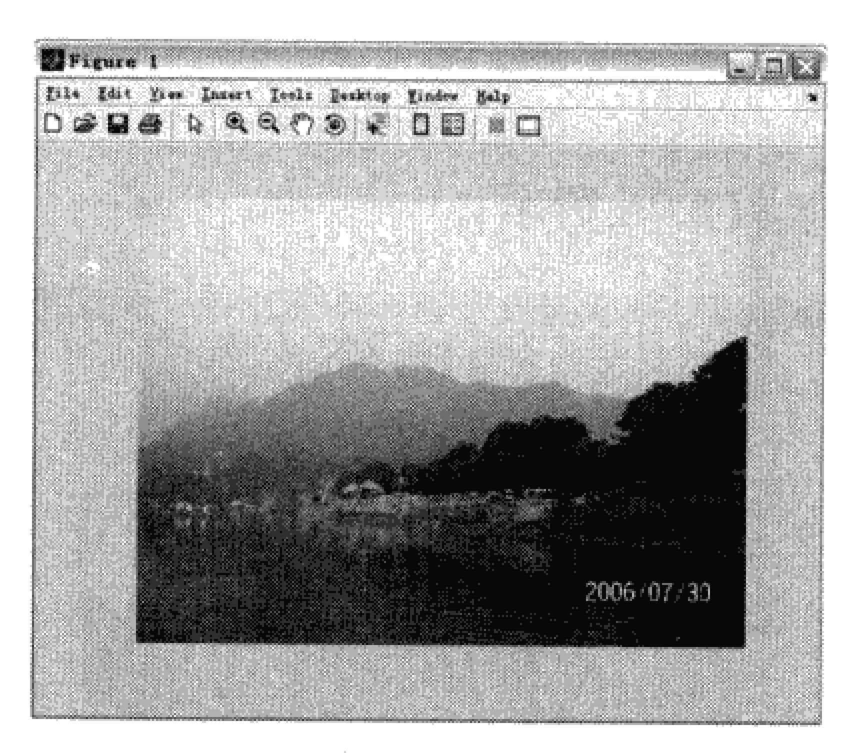


图 6.77　取消坐标轴后的图像显示

【实例讲解】 axis image off 进行了取消坐标轴的操作。

6.4.15　动画制作函数

如果将 MATLAB 产生的多幅图形保存起来，并利用系统提供的函数进行播放，就可产生动画效果。系统所提供的动画功能函数有 getframe、moviein 和 movie。

【语法说明】 m=getframe。

【功能介绍】 可将当前图形窗口作为一个画面取下并保存，它将每一帧画面信息数据截取下来整理成列向量。该函数截取图形的点阵信息，图形窗口的大小对数据向量的大小影响较大，窗口越大，所需存储容量越大。而图形的复杂性对数据容量要求没有直接的关系。

【语法说明】 m=moviein(n)。

【功能介绍】 用来建立一个足够大的 *n* 列的矩阵 *m*，用来保存 *n* 幅画面的数据，以备播放。

【实例 6.73】 播放一个不断变化的眼球程序段。

```
m=moviein(20);          %建立一个 20 个列向量组成的矩阵
for j=1:20
   plot(fft(eye(j+10))) %绘制出每一幅眼球图并保存到 m 矩阵中
   m(:,j)=getframe;
end
movie(m,10);             %以每秒 10 幅的速度播放画面
```

得到的图像如图 6.78 所示。

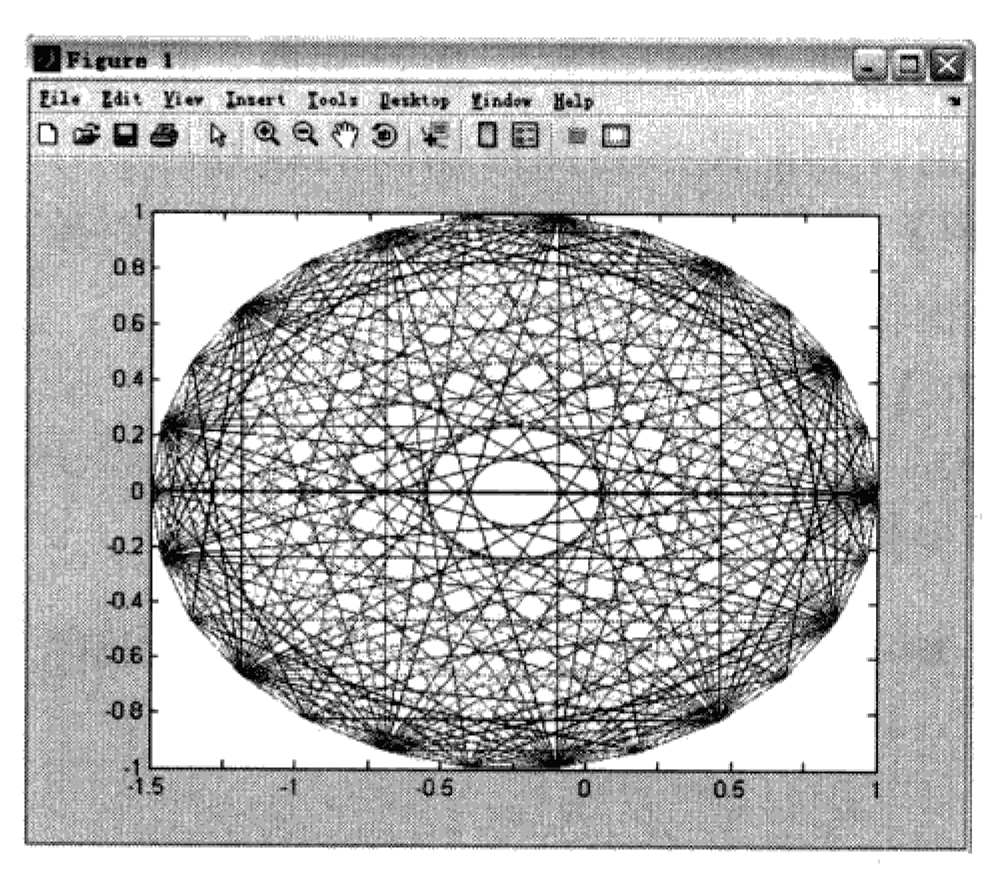


图 6.78 抓拍到的动画过程

【实例讲解】 建议用户实际在 MATLAB 中运行上面的函数，查看绘制过程。

6.5 图形句柄函数

图形句柄的操作函数对灵活地运用图形创造了很大的空间，利

用句柄的形式创建图形图像和绘制线条、面等极为方便。下面的小节将要对图形句柄命令进行讲解。

6.5.1 figure 函数——创建一个新的图形对象

【语法说明】

- figure：用默认的属性值创建一个新的图形对象。
- figure('PropertyName',PropertyValue, …)：对指定的属性 PropertyName 用指定的属性值 PropertyValue（属性名与属性值成对出现）创建一个新的图形窗口，对于那些没有指定的属性，则用默认值。
- figure(h)：若 h 为一个已经存在的图形的句柄，则 figure(h) 使由 h 标记的图形成为当前图形，使它可见，且在屏幕上把它显示到所有图形之前，当前图形为图像输出的地方。

【功能介绍】 创建一个新的图形对象。图形对象为在屏幕上单独的窗口，在窗口中可以输出图形。

【实例 6.74】 建立一个新的图形窗口。

```
>> figure
```

得到的图形如图 6.79 所示。

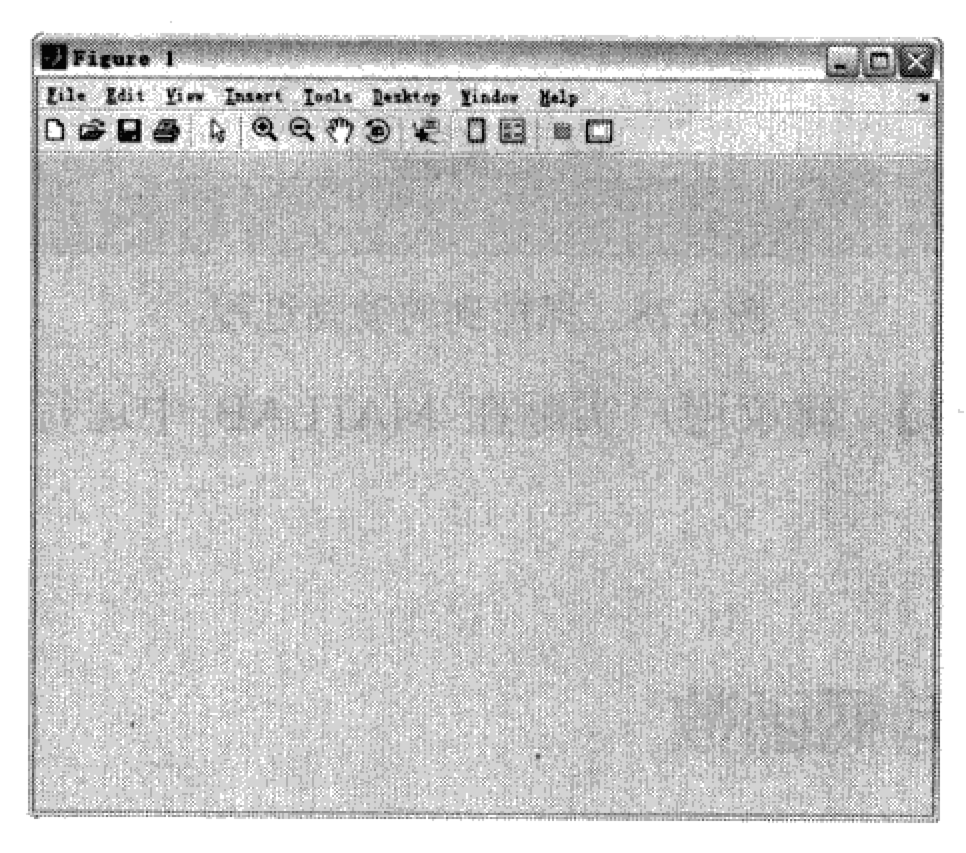


图 6.79　基本图形创建

【实例讲解】 通过这个简短的函数得到的就是一个基本的图形窗口，可以在这个基础上对图形进行绘制等操作。

6.5.2 line 函数——创建线条

【语法说明】

- line(X,Y)：在当前的坐标轴中画出由向量 X 和 Y 定义的线条。若 X 与 Y 为同型的矩阵，则对于 X、Y 的每一列画出一线条。
- line(X,Y,Z)：在三维空间中画出由 X、Y、Z 定义的线条。
- line(X,Y,Z,'PropertyName',PropertyValue,⋯)：画出由参数 X、Y、Z 确定的线条，其中对指定属性 PropertyName 设置为 PropertyValue，其他没有指定属性用默认值。
- line('PropertyName',PropertyValue,⋯)：对属性用相应的输入参数来设置而画出线条。这是函数 line 的低级使用形式，此时不接受矩阵参数。除了该情形，其他形式都接受矩阵参数。
- h = line(⋯)：返回每一条线的线对象对应的句柄向量。

【功能介绍】 生成线对象。函数 line 在当前坐标轴中生成一个线对象，可以指定线的颜色、宽度、类型和标记符号等其他特性。其中属性值和属性名称的对照表如表 6.1 所示。

表 6.1 线的属性名称与属性值对照表

属 性 名	说 明	有效属性值
	定义对象的数据	
Xdata	定义线条的 x 轴坐标参量	有效值：向量或矩阵 默认值：[0 1]
Ydata	定义线条的 y 轴坐标参量	有效值：向量或矩阵 默认值：[0 1]
Zdata	定义线条的 z 轴坐标参量	有效值：向量或矩阵 默认值：[0 1]

续表

属　性　名	说　　明	有效属性值
定义线型与数据点标记符		
LineStyle	定义线条的类型	有效值：-、--、:、-.、　none 默认值：-（实线）
LineWidth	定义线条的宽度（以磅为单位）	有效值：一标量 默认值：0.5 磅
Marker	定义标记数据点的标记符号	有效值：13 种类型之一 默认值：none
MarkerEdgeColor	定义标记颜色或可填充标记的边界颜色	有效值：auto、none、ColorSpec 默认值：auto
MarkerFaceColor	定义封闭形标记的填充颜色	
MarkerSize	定义标记大小	有效值：标量（磅） 默认值：6（磅）
控制线条的显示		
Clipping	坐标轴矩形区域是否可剪辑	有效值：on、off 默认值：on
EraseMode	定义显示与擦除线条的方法（对于动画显示）	有效值：normal、none、xor、background 默认值：normal
SelectionHighlight	当线条被选中时，是否突出显示	有效值：on、off 默认值：on
Visible	定义线条是否可见	有效值：on、off 默认值：on
Color	定义线条颜色	有效值：ColorSpec

续表

属 性 名	说 明	有效属性值
对象访问的控制		
HandleVisibility	定义线条句柄对其他函数是否可见	有效值：on、off、callback 默认值：on
HitTest	定义线条能否成为当前对象	有效值：on、off 默认值：on
关于线条的一般信息		
Children	线条没有子对象	有效值：[]（空矩阵）
Parent	线条对象的父对象为坐标轴对象	有效值：坐标轴句柄
Selected	是否显示线条的“选中”状态	有效值：on、off 默认值：on
Tag	用户定义的标签	有效值：任一字符串 默认值：''（空字符串）
Type	图形对象的类型（只读类型）	有效值：'line'
UserData	用户定义的数据	有效值：任一矩阵 默认值：[]（空矩阵）
与回调程序执行有关的属性		
BusyAction	定义如何处理回调中断程序	有效值：cancel、queue 默认值：queue
ButtonDownFcn	当在线条上按下鼠标时，定义一回调程序	有效值：字符串 默认值：' '（空字符串）
CreateFcn	当生成线条时，定义一回调程序	有效值：字符串 默认值：' '（空字符串）

续表

属　性　名	说　　明	有效属性值
DeleteFcn	当删除线条时，定义一回调程序	有效值：字符串 默认值：'　'（空字符串）
Interruptible	定义回调程序是否可中断	有效值：on、off 默认值：on（可中断）
UIContextMenu	定义与线条相关的菜单	有效值：UIContextMenu 的句柄

【实例 6.75】 生成函数 $y = e^x \times \sin(x)$ 的曲线，再画一条线条和颜色都稍作处理的这一函数。

```
>>t = 0:pi/20:2*pi;
>>hline1 = plot(t,exp(t).*sin(t),'k');
>>hline2 = line(t+.06,exp(t).*sin(t),'LineWidth',4,
Color',[.8 .8 .8]);
>>set(gca,'Children',[hline1 hline2])
```

得到的结果如图 6.80 所示。

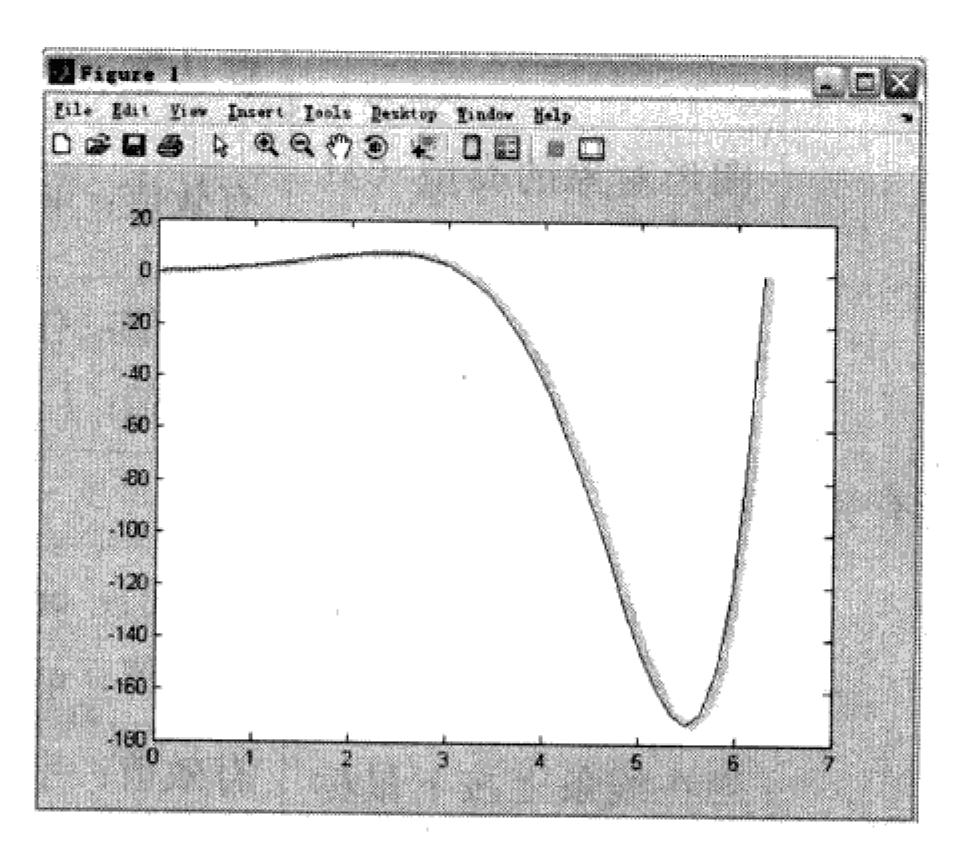


图 6.80　线条绘制

【实例讲解】 线条的 LineWidth 属性为 4，Color 属性由向量[.8 .8 .8]确定。

【实例 6.76】 用 line 画生成随机直线图。

```
>>line(rand(4,2),rand(4,2),rand(4,1))
>>line(rand(1,4),rand(1,4),rand(1,4))
>>line(rand(4,1),rand(4,1),rand(4,1))
>>line(rand(2,4),rand(2,4),rand(1,4))
>>line(rand(4,2),rand(4,2),rand(4,1))
```

得到的结果如图 6.81 所示。

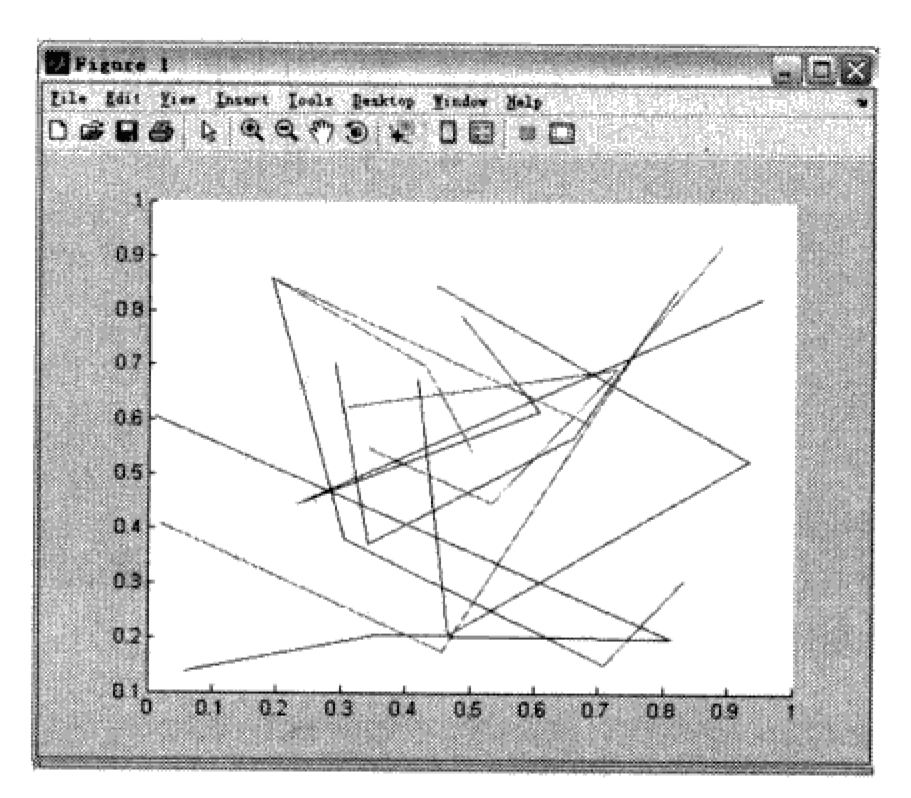


图 6.81　随机直线图

【实例讲解】 读者可以重新运行上面的函数，查看新的图形。

6.5.3　surface 函数——生成面

【语法说明】

- surface(Z)：画出由矩阵 Z 确定的曲面，其中 Z 为定义在一几何矩形区域上的单值函数。
- surface(Z,C)：画出颜色由 C 指定的、面由 Z 指定的空间曲面。
- surface(X,Y,Z)：曲面由参数 X、Y、Z 确定，颜色参数 C=Z，因此颜色能恰当地反映曲面的高度。
- surface(X,Y,Z,C)：曲面由参数 X、Y、Z 确定，颜色由参数 C 确定。
- surface(x,y,Z)：参数 x 与 y 为向量，若[m,n]=size(z)，则要

求 length(x)=n，length(y)=m，面上的点由(x(j),y(i),z(I,j))确定。

- surface(x,y,Z,C)：曲面确定如上情形，颜色由参数 C 确定。
- surface(…'PropertyName',PropertyValue,…)：对指定的曲面属性 PropertyName 指定为 PropertyValue，对曲面进行细微控制。

【功能介绍】 生成面对象。

【实例 6.77】 画出矩阵 $z=\begin{bmatrix}1 & 2 & 3\\4 & 7 & 9\\4 & 3 & 8\end{bmatrix}$ 确定的曲面。

```
>> z=[1 2 3;4 7 9;4 3 8]
z =
     1     2     3
     4     7     9
     4     3     8
>> surface(z)
>> colorbar('vert')
```

得到的结果如图 6.82 所示。

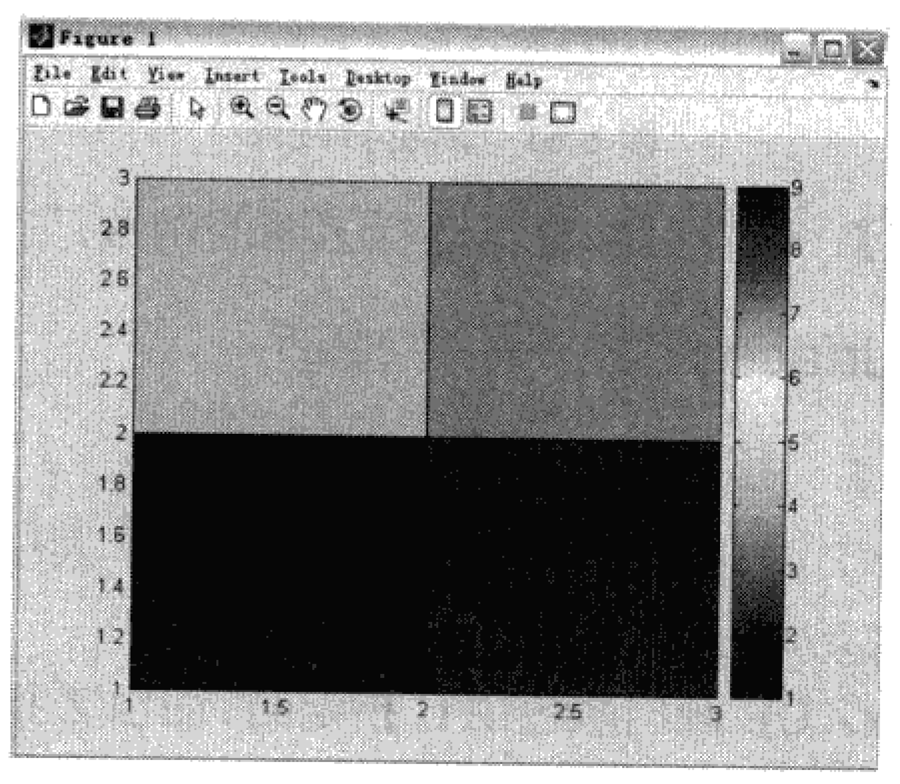
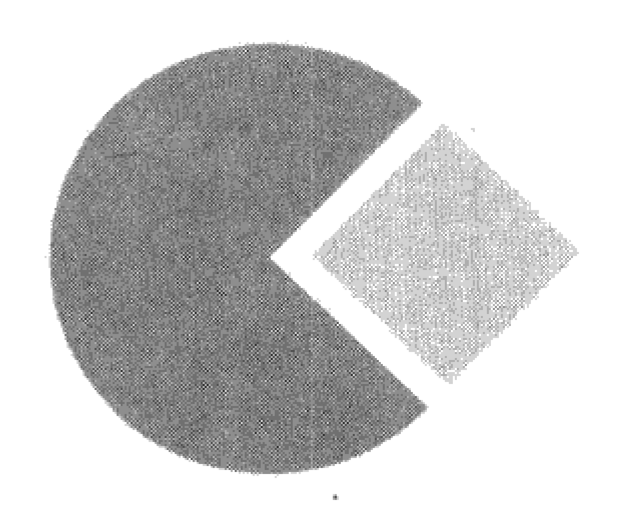


图 6.82　由矩阵确定曲面

【实例讲解】 图 6.82 所示为由矩阵 z 确定的曲面，在不同的区域上颜色的不同代表各个区域的数值不同，通过其右侧的刻度条可以得到每个区域的数值参量。



第7章 MATLAB 程序设计

为了灵活地运用 MATLAB 解决实际问题，充分调动 MATLAB 的科学资源，更深入地学习和利用这一仿真软件，有必要对 MATLAB 程序设计进行学习。MATLAB 不仅是一个功能强大的工具软件，更是一种高效的编程语言。MATLAB 软件环境包括了 MATLAB 语言的编程环境，M 文件则是用 MATLAB 语言编写的程序代码文件。

7.1 MATLAB 程序入门简介

在这一节中，将要通过几个例子来介绍 MATLAB 程序的编写、调试和运行情况。通过最基本的操作介绍 MATLAB 程序，为后面的学习奠定基础。

7.1.1 MATLAB 文本编辑器

M 文件是一个文本文件，它可以用任何编辑程序来建立和编辑，而一般常用且最为方便的是使用 MATLAB 提供的文本编辑器。

MATLAB 文本编辑器是用来编写 MATLAB 程序的环境，如图 7.1 所示。在这个环境下可以随时修改和运行程序。

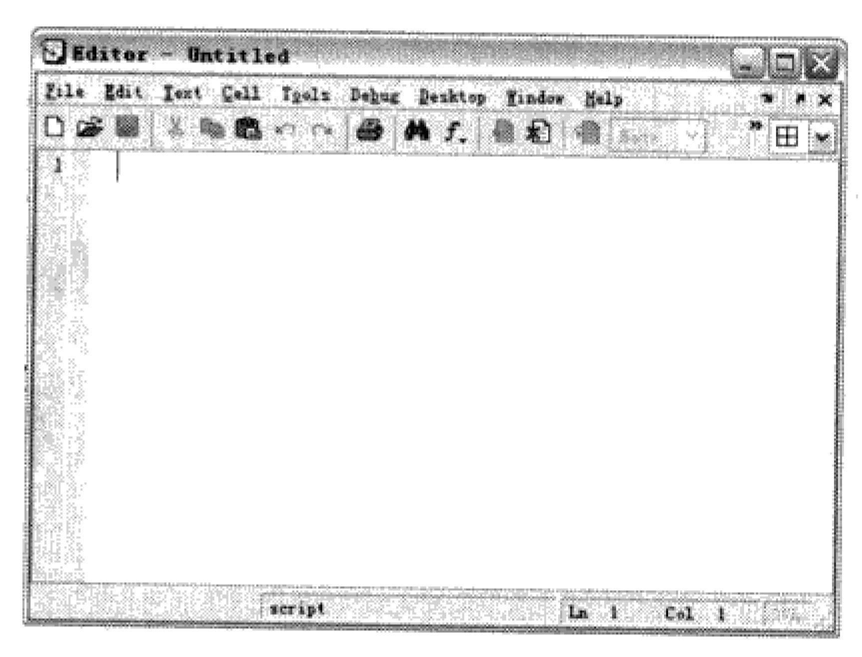


图 7.1 MATLAB Editor/Debugger 窗口

为建立新的 M 文件，启动 MATLAB 文本编辑器有以下 3 种方法。

- 菜单操作。在 MATLAB 主窗口中选择“File” | “New” | “M-file”命令，屏幕上将出现 MATLAB 文本编辑器窗口。
- 命令操作。在 MATLAB 命令窗口中输入命令 edit，启动 MATLAB 文本编辑器后，输入 M 文件的内容并存盘。
- 命令按钮操作。单击 MATLAB 主窗口工具栏上的 New M-File 命令按钮，启动 MATLAB 文本编辑器后，输入 M 文件的内容并存盘。

打开已有的 M 文件，也有 3 种方法。

- 菜单操作。选择 MATLAB 主窗口的“File” | “Open”命令，则屏幕出现 Open 对话框，在 Open 对话框中选中所需打开的 M 文件。在文档窗口可以对打开的 M 文件进行编辑修改，编辑完成后，将 M 文件存盘。
- 命令操作。在 MATLAB 命令窗口输入命令：edit 文件名，则打开指定的 M 文件。
- 命令按钮操作。选择 MATLAB 主窗口工具栏上的“Open File”命令按钮，再从弹出的对话框中选择所需打开的 M 文件。

在空白区域中可以编写自己的 MATLAB 程序，在这个编辑窗口有需要的基本命令。如果要保存这个文件，选择“File”|“Save As”命令，如图 7.2 所示。

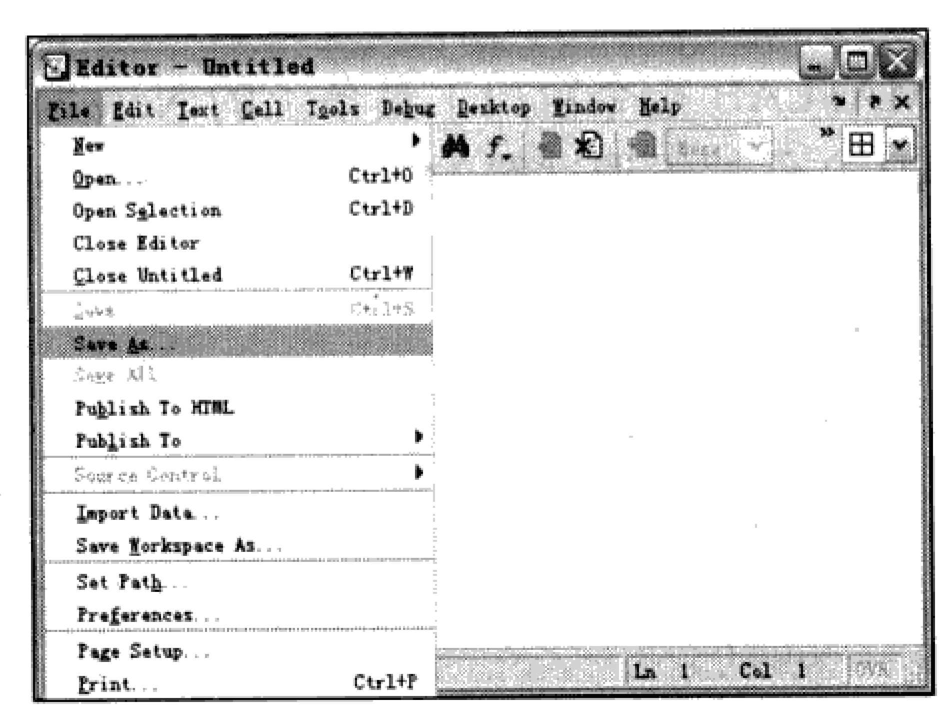


图 7.2 M 文件的保存

如果要运行编辑好的 M 程序文件，那么选择工具栏中的“Debug”|“Run”命令，如图 7.3 所示。

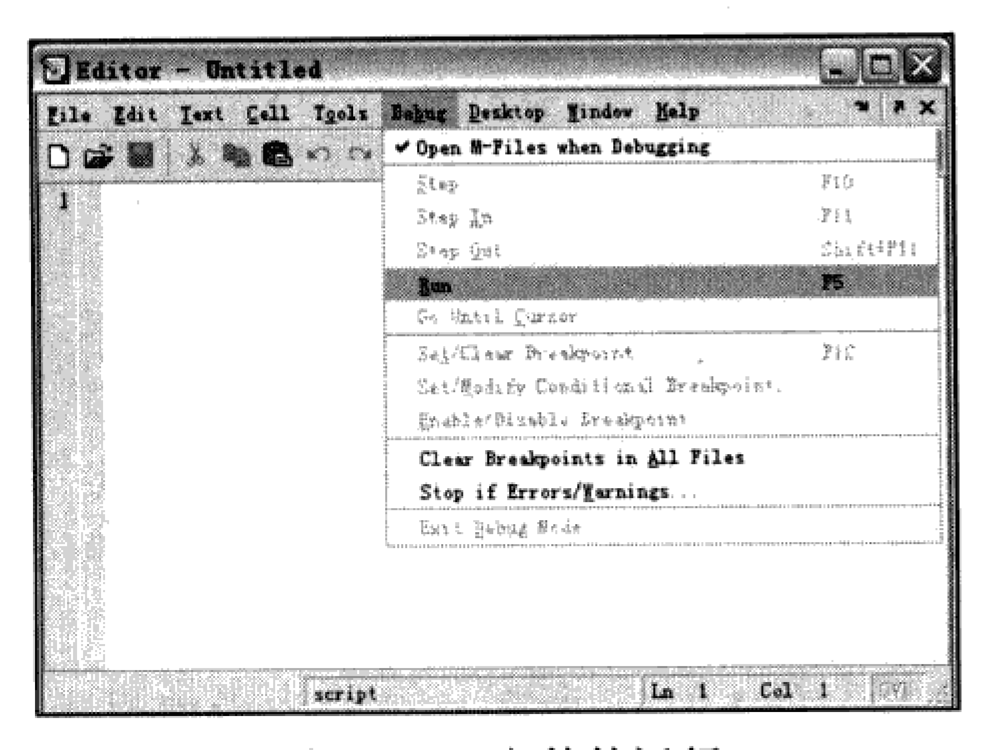


图 7.3 M 文件的运行

7.1.2 利用文本编辑器编写 M 文件

【实例 7.1】 通过 M 脚本文件，画出下列分段函数所表示的曲面。

$$p(x_1,x_2)=\begin{cases}0.5457\mathrm{e}^{-0.75x_2^2-3.75x_1^2-1.5x_1} & x_1+x_2>1\\0.7575\mathrm{e}^{-x_2^2-6x_1^2} & -1<x_1+x_2\leqslant 1\\0.5457\mathrm{e}^{-0.75x_2^2-3.75x_1^2+1.5x_1} & x_1+x_2\leqslant -1\end{cases}$$

（1）单击 MATLAB 指令窗口工具条上的 New File 图标，就可打开 MATLAB 文件编辑调试器 MATLAB Editor/Debugger。其窗口名为 untitled。用户即可在空白窗口中编写程序。

（2）输入如下一段程序：

```
a=2;b=2;                                          %              <2>
clf;
x=-a:0.2:a;y=-b:0.2:b;
for i=1:length(y)
   for j=1:length(x)
      if x(j)+y(i)>1
         z(i,j)=0.5457*exp(-0.75*y(i)^2-3.75*x(j)^2-
1.5*x(j));
      elseif x(j)+y(i)<=-1
         z(i,j)=0.5457*exp(-0.75*y(i)^2-3.75*x(j)^2+
1.5*x(j));
      else z(i,j)=0.7575*exp(-y(i)^2-6.*x(j)^2);
      end
   end
end
axis([-a,a,-b,b,min(min(z)),max(max(z))]);
colormap(flipud(winter));surf(x,y,z);
```

（3）单击编辑调试器工具条图标，在弹出的“保存为”对话框中，选择“保存文件夹”，键入新编文件名（如 L91.m），单击【保存】按钮，就完成了文件保存。

（4）运行文件。使 L91.m 所在目录成为当前目录，或让该目录处在 MATLAB 的搜索路径上。然后运行指令“L91”，便可得到如图 7.4 所示图形。

【实例讲解】 从上面的结果可以看出，直接运行 M 文件名称就可以得到结果。

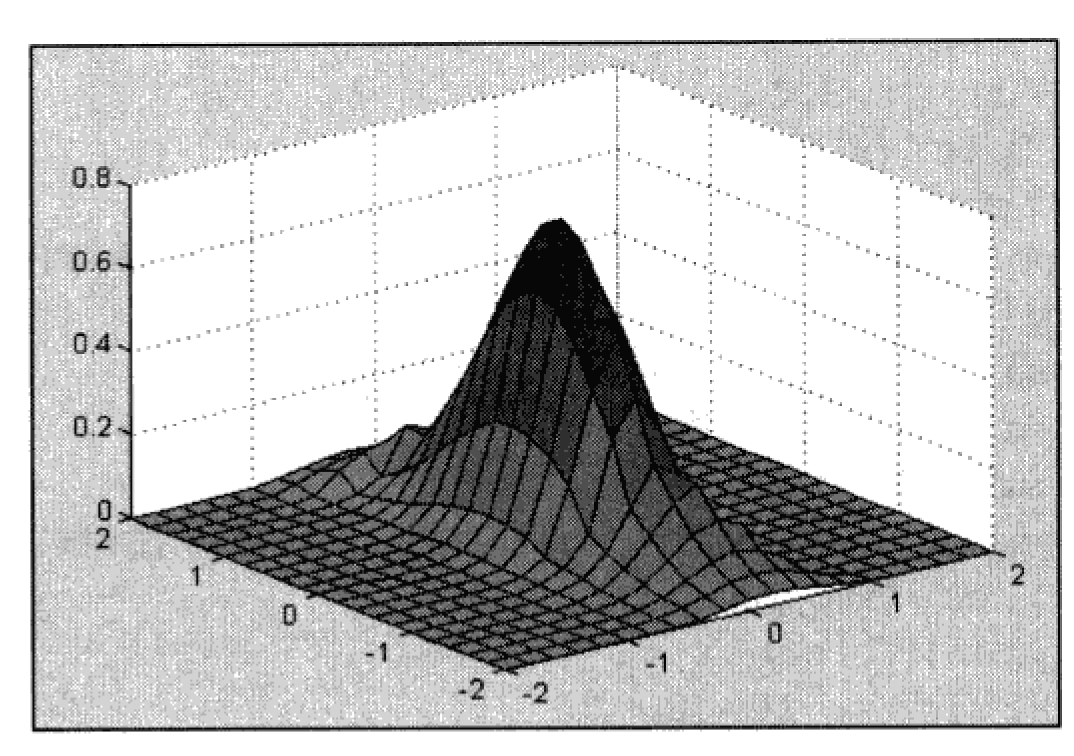


图 7.4　运行 L91.m 得到的图形

【实例 7.2】 建立一个绘制不同函数曲线的用户界面。

首先建立命令文件，程序代码如下：

```
h0=figure('toolbar','none',...          %图形界面的建立
    'position',[198 56 408 468],...
    'name','实例 01');
h1=axes('parent',h0,...                 %坐标轴设置
    'position',[0.29 0.45 0.7 0.5],...
    'visible','on');
f=uicontrol('parent',h0,...             %框架界面
    'style','frame',...
    'position',[5 50 90 400]);
p1=uicontrol('parent',h0,...            %建立按钮控件
    'style','pushbutton',...
    'position',[150 100 60 40],...
    'string','绘图',...
    'callback',[...
        'm=str2num(get(e1,''string''));,',...
        'n=str2num(get(e2,''string''));,',...
        'a=get(l1,''value'');,',...
        'x=m:0.1:n;',...
        'if a==1,',...
        'plot(x,sin(x)),',...
        'end,',...
        'if a==2,',...
        'plot(x,cos(x)),',...
```

```
            'end,',...
            'if a==3,',...
            'plot(x,exp(x)),',...
            'end']);
   p2=uicontrol('parent',h0,...     %建立[选择函数]按钮控件
       'style','pushbutton',...
       'position',[270 100 60 40],...
       'string','选择函数',...
       'callback','close');
   l1=uicontrol('parent',h0,...     %创建下拉列表框,下拉菜单分别有sin(x),cos(x),exp(x)
       'style','listbox',...
       'position',[10 300 80 80],...
       'string','sin(x)|cos(x)|exp(x)',...
       'value',1,...
       'max',0.5,...
       'min',0);
   f2=uicontrol('parent',h0,...     %创建文本框
       'style','text',...
       'string','选择函数',...
       'fontsize',10,...
       'position',[10 380 80 20]);
   r1=uicontrol('style','radio',...
       'string','grid on',...
       'value',0,...
       'position',[10 100 60 20],...
       'callback',[...
          'grid on,',...
          'set(r1,''value'',1);,',...
          'set(r2,''value'',0)']);
   r2=uicontrol('style','radio',...
       'string','grid off',...
       'position',[10 80 60 20],...
       'value',1,...
       'callback',[...
          'grid off,',...
          'set(r2,''value'',1);,',...
          'set(r1,''value'',0)']);
   e1=uicontrol('parent',h0,...     %创建可编辑文本框
       'style','edit',...
       'string',0,...
```

```
    'position',[20 210 60 20],...
    'horizontalalignment','right');
e2=uicontrol('parent',h0,...
    'style','edit',...
    'string','3',...
    'position',[20 150 60 20],...
    'horizontalalignment','right');
t1=uicontrol('parent',h0,...
    'style','text',...
    'string','X from',...
    'fontsize',10,...
    'position',[20 230 60 20],...
    'horizontalalignment','center');
t2=uicontrol('parent',h0,...
    'style','text',...
    'string','To',...
    'fontsize',10,...
    'position',[20 170 60 20],...
    'horizontalalignment','center');
```

通过上面的程序代码得到的图形界面如图7.5所示。

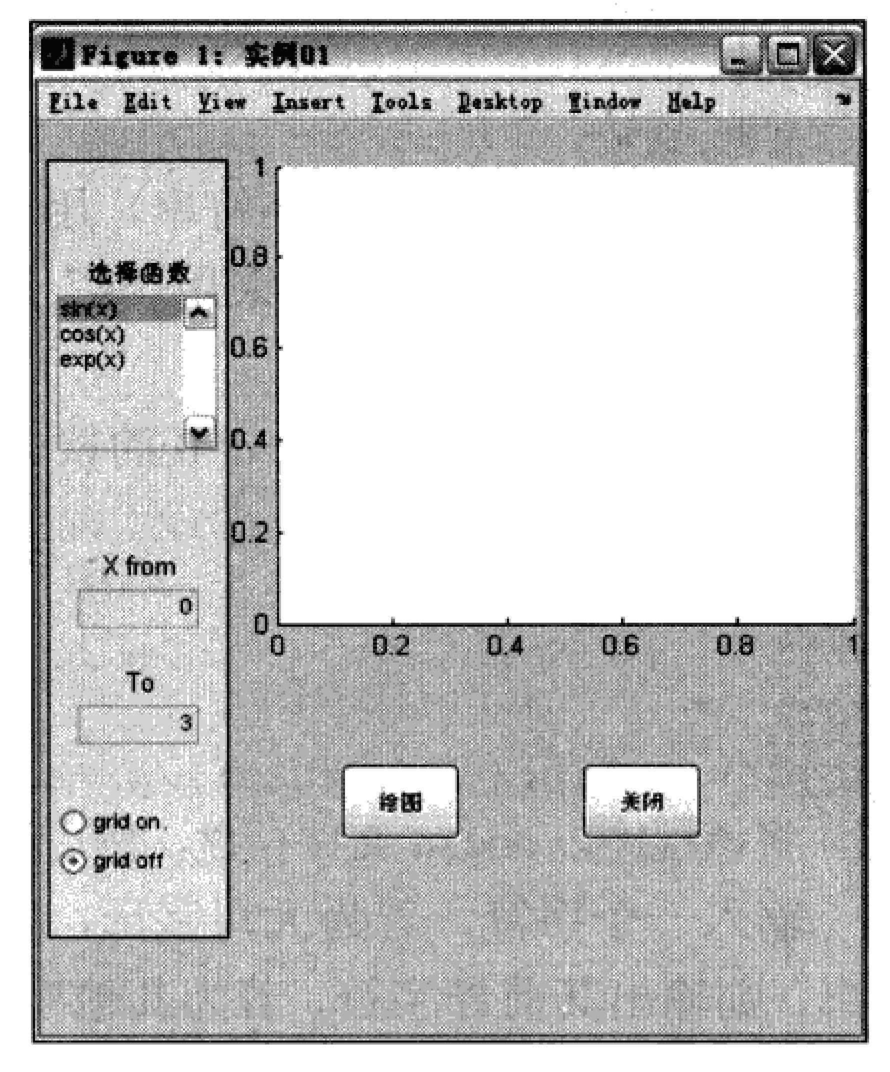


图7.5　图形界面设计代码

【实例讲解】 figure 函数用来建立图形界面，uicontrol 用来建立界面上的各种各样的控件。

7.2 MATLAB 控制流

计算机编程语言提供许多功能，它使用户根据决策结构控制命令执行流程。如果以前已经使用过这些功能，对此就会很熟悉。控制流非常重要，因为它使过去的计算影响将来的运算。MATLAB 提供 3 种决策或控制流结构，它们是 For 循环、While 循环和 If-Else-End 结构。这些结构经常包含大量的 MATLAB 命令，所以经常出现在 M 文件中，而不是直接加在 MATLAB 命令行。

7.2.1 input 函数——数据的输入

【语法说明】 从键盘输入数据，则可以使用 input 函数来进行，该函数的调用格式为：

```
A=input(提示信息,选项)
```

【功能介绍】 其中提示信息为一个字符串，用于提示用户输入什么样的数据。如果在 input 函数调用时采用“s”选项，则允许用户输入一个字符串。

【实例 7.3】 求一元二次方程 $a^2+bx+c=0$ 的根。

```
a=input('a=?');
b=input('b=?');
c=input('c=?');
d=b*b-4*a*c;
x=[(-b+sqrt(d))/(2*a),(-b-sqrt(d))/(2*a)]
```

【实例讲解】 将该程序以 abc.m 文件存盘，然后运行 abc.m 文件。

【实例 7.4】 输入一个字符，若为大写字母，则输出其后继字符，若为小写字母，则输出其前导字符，若为其他字符则原样输出。新建一个 M 文件，如下：

```
c=input('','s');
  if c>='A' & c<='Z'
     disp(setstr(abs(c)+1));
    else if c>='a'& c<='z'
     disp(setstr(abs(c)-1));
    else
       disp(c);
    end
```

【实例讲解】 disp 为输出函数，会在后面的小节中讲解，

7.2.2　disp 函数——数据的输出

【语法说明】 MATLAB 提供的命令窗口输出函数主要有 disp 函数，其调用格式为：

```
disp(输出项)
```

【功能介绍】 其中输出项既可以为字符串，也可以为矩阵。

【实例 7.5】 输入 x，y 的值，并将它们的值互换后输出。

```
>> x=input('Input x please.');
Input x please.4                  %4 为键盘输入的值
>> y=input('Input y please.');
Input y please.6                  %6 为键盘输入的值
>> z=x;
>> x=y;
>> y=z;
>> disp(x);
    6
>> disp(y);
    4
```

【实例讲解】 分别输入 x，y 的值，而后输出，这个例子中原来的 x=4，y=6，互换后 x=6，y=4。

【实例 7.6】 求一元二次方程 $4x^2+20x+6=0$ 的根。

```
>> a=4
a =
    4
>> b=20
b =
   20
>> c=6
```

```
c =
     6
>> d=b*b-4*a*c
d =
   304
>> x=[(-b+sqrt(d))/(2*a),(-b-sqrt(d))/(2*a)]
x =
   -0.3206   -4.6794

>> disp(['x1=',num2str(x(1)),',x2=',num2str(x(2))])
x1=-0.32055,x2=-4.6794
```

【实例讲解】 分别输入 *x*，*y* 的值，进行数值计算之后输出。

7.2.3　pause 函数——程序的暂停

【语法说明】 暂停程序的执行可以使用 pause 函数，其调用格式为：

```
pause(延迟秒数)
```

【功能介绍】 如果省略延迟时间，直接使用 pause，则将暂停程序，直到用户按任一键后程序继续执行。若要强行中止程序的运行可使用 Ctrl+C 命令。

7.2.4　for 循环

【语法说明】 for 循环允许一组命令以固定的和预定的次数重复。for 循环的一般形式是：

```
for 循环变量=表达式 1:表达式 2:表达式 3
   循环体语句
end
```

【功能介绍】 用来循环一组或一个 MATLAB 命令。其中表达式 1 的值为循环变量的初值，表达式 2 的值为步长，表达式 3 的值为循环变量的终值。步长为 1 时，表达式 2 可以省略。

【实例 7.7】 一个简单的 for 循环示例。

```
for i=1:10;             %i 依次取 1,2,…10,.
x(i)=i;              %对每个 i 值,重复执行由该指令构成的循环体
end;
```

```
x                          %要求显示运行后数组 x 的值
x =
     1     2     3     4     5     6     7     8     9    10
```

【实例讲解】 在上面的结果中，没有设置其他的结束条件。

【实例 7.8】 求 sin(*n**pi/10)的值，其中 *n*=1,2,…,10。

```
» for n=1:10
    x(n)=sin(n*pi/10);
 end

» x
x =
  Columns 1 through 7
    0.3090 0.5878 0.8090 0.9511 1.0000 0.9511  0.8090
  Columns 8 through 10
    0.5878    0.3090    0.0000
```

【实例讲解】 第一语句是说：对 *n* 等于 1～10，求所有语句的值，直至下一个 end 语句。第一次通过 For 循环 n=1，第二次，n=2，如此继续，直至 n=10。在 n=10 以后，For 循环结束，然后求 end 语句后面的任何命令值，在这种情况下显示所计算的 *x* 的元素。

【实例 7.9】 For 循环不能用 For 循环内重新赋值循环变量 *n* 来终止。

```
» for n=1:10
    x(n)=sin(n*pi/n);
    n=10;
 end

» x
x =
  Columns 1 through 7
    0.3090  0.5878 0.8090 0.9511 1.0000 0.9511  0.8090
  Columns 8 through 10
    0.5878    0.3090    0.0000
```

【实例讲解】 从程序结果中可以看出，程序中的“n=10”是无效的。

【实例 7.10】 语句 1:10 是一个标准的 MATLAB 数组创建语句。在 For 循环内接受任何有效的 MATLAB 数组。

```
» data=[3  9  45  6;  7  16  -1  5]
data =
     3     9    45     6
     7    16    -1     5

for n=data
          x=n(1)-n(2)
       end
x =
    -4
x =
    -7
x =
    46
x =
     1
```

【实例讲解】 从上面的结果中可以看出，程序计算出数组两行对应数据相减的结果。

【实例 7.11】 For 循环可按需要嵌套。

```
for n=1:5
   for m=5:-1:1
     A(n,m)=n^2+m^2;
   end
   disp(n)
  end
     1
     2
     3
     4
     5

» A
A =
     2     5    10    17    26
     5     8    13    20    29
    10    13    18    25    34
    17    20    25    32    41
    26    29    34    41    50
```

【实例讲解】 在上面的例子中，用户使用了两个循环变量来实

现嵌套。

【实例 7.12】 实际应用 for 函数绘图：绘制 $y = 1 - \frac{1}{\beta} e^{-\varepsilon t} \sin(\beta t + \theta)$，$\beta = \sqrt{1 - \varepsilon^2}$，$\theta = \arctan \frac{\sqrt{1 - \varepsilon^2}}{\varepsilon}$，$\varepsilon = 0.2, 0.4, 0.6, 0.8$，$t = [0, 16]$ 的曲线。

建立 M 文件如下所示：

```
clf;
t=[0:0.1:16]';

for x=0.2:0.2:0.8
    b=sqrt([1-x*x]);
    z=atan(b/x);
    y1=-t*x;
    y2=t*b+z;
    y=1-exp(y1).*sin(y2)/b;
    plot(t,y),hold on
end
```

得到的图形如图 7.6 所示。

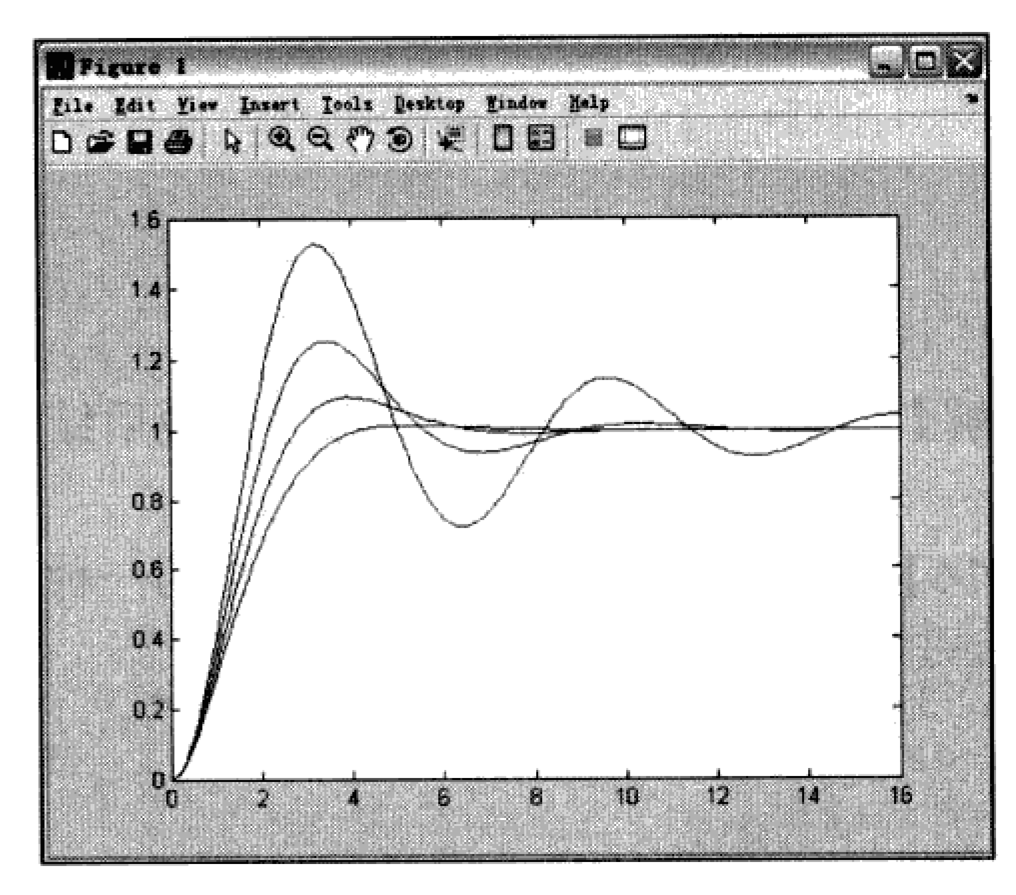


图 7.6 二阶系统阶跃响应图

【实例讲解】 利用循环语句控制初始值和结束值，以及步长，并在循环体中实现绘点的操作。

7.2.5 while 循环

【语法说明】 与 for 循环以固定次数求一组命令的值不同，while 循环以不固定的次数求一组语句的值。while 循环的一般形式是：

```
while expression
{commands}
end
```

【功能介绍】 只要在表达式里的值为真，就执行 while 和 end 语句之间的{commands}。通常，表达式的求值给出一个标量值，数组值也同样有效。在数组情况下，所得到数组的所有元素必须都为真。

【实例 7.13】 Fibonacci 数组的元素满足 Fibonacci 规则：$a_{k+2}=a_k+a_{k+1}$，$(k=1,2,\cdots)$；且 $a_1=a_2=1$。现要求该数组中第一个大于 10000 的元素。

```
a(1)=1;a(2)=1;i=2;
while  a(i)<=10000
    a(i+1)=a(i-1)+a(i); %当现有的元素仍小于 10000 时，求解
下一个元素
    i=i+1;
end;
i,a(i),
i =
    21
ans =
  10946
```

【实例讲解】 这是在 M 文件中的操作，最后结果在命令行中显示。

【实例 7.14】 计算特殊 MATLAB 值 EPS 的一种方法。

```
» num=0;EPS=1;
» while (1+EPS)>1
    EPS=EPS/2;
    num=num+1;
  end
```

```
» num
num =
    53

» EPS=2*EPS
EPS =
  2.2204e-016
```

【实例讲解】 这个例子表明：它是一个可加到 1，而使结果以有限精度大于 1 的最小数值。这里用大写 EPS 表示，因此 MATLAB 的 EPS 的值不会被覆盖掉。在该例子中，EPS 以 1 开始。只要 (1+EPS)>1 为真（非零），就一直求 While 循环内的命令值。由于 EPS 不断地被 2 除，EPS 逐渐变小以至到 EPS+1 不大于 1。

【实例 7.15】 从键盘输入若干个数，当输入 0 时结束输入，求这些数的平均值和它们之和。

```
sum=0;
cnt=0;
val=input('Enter a number (end in 0):');
while (val~=0)
     sum=sum+val;
     cnt=cnt+1;
     val=input('Enter a number (end in 0):');
end
if (cnt > 0)
    sum
    mean=sum/cnt
end
```

【实例讲解】 用户可以自行演示上面的程序代码，查看运行的结果。

7.2.6 if-else-end 结构控制语句

【语法说明】 命令的序列必须根据关系的检验有条件地执行。在编程语言里，这种逻辑由某种 if-else-end 结构来提供。最基本的 if-else-end 结构是：

```
    if  条件 1
        语句组 1
```

```
    elseif  条件 2
        语句组 2
        ……
    elseif  条件 m
          语句组 m
    else
          语句组 n
    end
```

【功能介绍】 语句用于实现多分支选择结构。如果在表达式中的所有元素为真（非零），那么就执行 if 和 end 语言之间的{commands}。在表达式包含有几个逻辑子表达式时，即使前一个子表达式决定了表达式的最后逻辑状态，仍要计算所有的子表达式。

【实例 7.16】 一个简单的分支结构。

```
cost=10;number=12;
if number>8
   sums=number*0.95*cost;
end,sums
  sums =
  114.0000
```

【实例讲解】 首先给 cost 和 number 两个变量赋值，由于 12>8，所以，这个分支语句被执行，得到了 sums 的值，如果 number 的值小于或等于 8，sums 为 0。

【实例 7.17】 用 for 循环指令来寻求 Fibonacc 数组中第一个大于 10000 的元素。

```
n=100;a=ones(1,n);
for i=3:n
     a(i)=a(i-1)+a(i-2);
     if a(i)>=10000
        a(i),
        break;  %跳出所在的一级循环
     end;
end,i
ans =
       10946
i =
21
```

【实例讲解】 如果数组 a 中有某个元素的值大于 10000，先将这个数值输出，然后跳出循环，中断执行。

【实例 7.18】 计算一个简单的数学乘法问题。

```
» apples=10;          % number of apples
» cost=apples*25      % cost of apples
cost =
   250

» if apples>5 % give 20% discount for larger purchases
    cost=(1-20/100)*cost;
 end

» cost
cost =
   200
```

【实例讲解】 上面的 if 结构是最简单的结构，在后面的例子中将演示复杂 if 结构的用法。

【实例 7.19】 输入一个字符，若为大写字母，则输出其对应的小写字母；若为小写字母，则输出其对应的大写字母；若为数字字符则输出其对应的数值，若为其他字符则原样输出。

```
c=input('请输入一个字符','s');
if c>='A' & c<='Z'
   disp(setstr(abs(c)+abs('a')-abs('A')));
elseif c>='a'& c<='z'
    disp(setstr(abs(c)- abs('a')+abs('A')));
elseif c>='0'& c<='9'
    disp(abs(c)-abs('0'));
else
    disp(c);
end
```

【实例讲解】 用户可以自行检测上面的程序代码，查看运行的结果。

7.2.7 switch-case 结构

【语法说明】 switch-case 结构为多种选择的分支结构，在这个

结构中，switch 语句根据表达式的取值不同，分别执行不同的语句，其语句格式为：

```
switch  表达式
   case  表达式 1
      语句组 1
   case  表达式 2
      语句组 2
      ......
   case  表达式 m
      语句组 m
   otherwise
      语句组 n
End
```

【功能介绍】 用来编写有多种选择情况下的程序。当表达式的值等于表达式 1 的值时，执行语句组 1；当表达式的值等于表达式 2 的值时，执行语句组 2；……；当表达式的值等于表达式 m 的值时，执行语句组 m；当表达式的值不等于 case 所列的表达式的值时，执行语句组 n。当任意一个分支的语句执行完后，直接执行 switch 语句的下一句。

【实例 7.20】 学生的成绩管理，用来演示 switch 结构的应用。

```
clear;
%划分区域：满分(100),优秀(90-99),良好(80-89),及格(60-79),不及格(<60)。
for i=1:10;a{i}=89+i;b{i}=79+i;c{i}=69+i;d{i}=59+i; end;
c=[d,c];
Name={' Jack','Marry','Peter','Rose',' Tom'};%元胞数组
Mark={72,83,56,94,100};Rank=cell(1,5);
%创建一个含 5 个元素的构架数组 S,它有三个域。
S=struct('Name',Name,'Marks',Mark,'Rank',Rank);
%根据学生的分数,求出相应的等级。
for i=1:5
   switch S(i).Marks
   case 100          %得分为 100 时
      S(i).Rank='满分';        %列为'满分'等级
   case a            %得分在 90 和 99 之间
      S(i).Rank='优秀';        %列为'优秀'等级
   case b            %得分在 80 和 89 之间
      S(i).Rank='良好';        %列为'良好'等级
   case c            %得分在 60 和 79 之间
```

```
        S(i).Rank='及格';      %列为'及格'等级
     otherwise        %得分低于60
        S(i).Rank='不及格';    %列为'不及格'等级
     end
  end
  %将学生姓名,得分,登记等信息打印出来。
  disp(['学生姓名  ','  得分  ','    等级']);disp(' ')
  for i=1:5;
   disp([S(i).Name,blanks(6),num2str(S(i).Marks),blanks
(6),S(i).Rank]);
  end;
  学生姓名    得分      等级

   Jack      72       及格
  Marry      83       良好
  Peter      56      不及格
   Rose      94       优秀
    Tom     100       满分
```

【实例讲解】 这个例子就是根据学生的分数来确定等级，因为分数在0～100之间不等，所以有多种选择，而分数的阶段限制与不同的等级是对应的。所以类似这样的情况，用这个语句是合理的。

【实例7.21】 某商场对顾客所购买的商品实行打折销售，标准如下（商品价格用price来表示）：

```
     price<200            没有折扣
     200≤price<500       3%折扣
     500≤price<1000      5%折扣
     1000≤price<2500     8%折扣
     2500≤price<5000     10%折扣
  5000≤price          14%折扣
```

输入所售商品的价格，求其实际销售价格。

```
  price=input('请输入商品价格');
  switch fix(price/100)
     case {0,1}                  %价格小于200
        rate=0;
     case {2,3,4}                %价格大于等于200但小于500
        rate=3/100;
     case num2cell(5:9)          %价格大于等于500但小于1000
        rate=5/100;
     case num2cell(10:24)        %价格大于等于1000但小于2500
```

```
      rate=8/100;
   case num2cell(25:49)      %价格大于等于2500但小于5000
      rate=10/100;
   otherwise                 %价格大于等于5000
      rate=14/100;
end
price=price*(1-rate)                  %输出商品实际销售价格
得到的结果:
请输入商品价格 69

price =

    69

请输入商品价格 247

price =

  239.5900

请输入商品价格 3980

price =

        3582
```

【实例讲解】 程序运行后会在 MATLAB 的命令行中出现“请输入商品价格”的字样，人为输入价格后，就会得到打折后的价格。

7.2.8 try-catch 结构

【语法说明】 try 语句先试探性执行语句组 1，如果语句组 1 在执行过程中出现错误，则将错误信息赋给保留的 lasterr 变量，并转去执行语句组 2。

```
try
   语句组 1
catch
   语句组 2
end
```

【实例 7.22】 try-catch 结构应用实例。

```
clear,N=4;A=magic(3);%设置 3 行 3 列矩阵 A
try
  A_N=A(N,:),      %取 A 的第 N 行元素
catch
    A_end=A(end,:),%如果取 A(N,:)出错,则改取 A 的最后一行
end
lasterr     %显示出错原因
A_end =
    4     9     2
ans =
 Index exceeds matrix dimensions.
```

【实例讲解】 在上面的例子中，“Index exceeds matrix dimensions”就是显示的错误信息。

【实例 7.23】 矩阵乘法运算要求两矩阵的维数相容，否则会出错。先求两矩阵的乘积，若出错，则自动转去求两矩阵的点乘。

```
A=[1,2,3;4,5,6]; B=[7,8,9;10,11,12];
try
  C=A*B;
catch
  C=A.*B;
end
C
lasterr                 %显示出错原因
```

【实例讲解】 上例显示出错信息。

7.2.9 在 M 文件中使用控制流

【实例 7.24】 M 函数文件示例。

```
[circle.m]
function  sa = circle(r,s)
%CIRCLE   plot a circle of radii r in the line specified
by s.
%    r 指定半径的数值
%    s 指定线色的字符串
%    sa 圆面积
%
% circle(r)利用蓝实线画半径为 r 的圆周线
% circle(r,s)利用串 s 指定的线色画半径为 r 的圆周线
% sa=circle(r)          计算圆面积,并画半径为 r 的蓝色圆面
```

```
% sa=circle(r,s)      计算圆面积,并画半径为 r 的 s 色圆面

if nargin>2
   error('输入变量数太多。');
end;
if nargin==1
   s='b';
end;
clf;
t=0:pi/100:2*pi;
x=r*exp(i*t);
if nargout==0
   plot(x,s);
else
   sa=pi*r*r;
   fill(real(x),imag(x),s)
end
axis('square')
```

【实例讲解】 这个例子程序用来画半径为 r 的圆形，颜色由 s 字符串来指定。

7.2.10 continue 语句

【语法说明】 continue;

【功能介绍】 一般与 if 语句配合使用。

continue 语句控制跳过循环体中的某些语句。当在循环体内执行到该语句时，程序将跳过循环体中所有剩下的语句，继续下一次循环。

【实例 7.25】 求[100，200]之间第一个能被 21 整除的整数。

```
for n=100:200
if rem(n,21)~=0
     continue
end
break
end
n
```

【实例讲解】 这个例子中，执行到 continue 时只是跳过本次循环，下一次的循环还会继续，而 break 则是跳出整个 for 循环。

【实例 7.26】 建立一个简单的循环，演示 continue 命令的用法。

```
function 11
for j=1:10
    if j==6
        continue
    end
    fprintf('j=%d\n',j);
```

得到的结果为：

```
ans =
    11
```

【实例讲解】 continue 不会终止程序的执行。

【实例 7.27】 利用 M 文件绘制正弦曲线图。

```
function shili01
h0=figure('toolbar','none',...
    'position',[198 56 350 300],...
    'name','实例 01');
h1=axes('parent',h0,...
   'visible','off');
x=-pi:0.05:pi;
y=sin(x);
plot(x,y);
xlabel('自变量 X');
ylabel('函数 Y');
title('SIN ( ) 三角函数');
grid on
```

得到的图形如图 7.7 示。

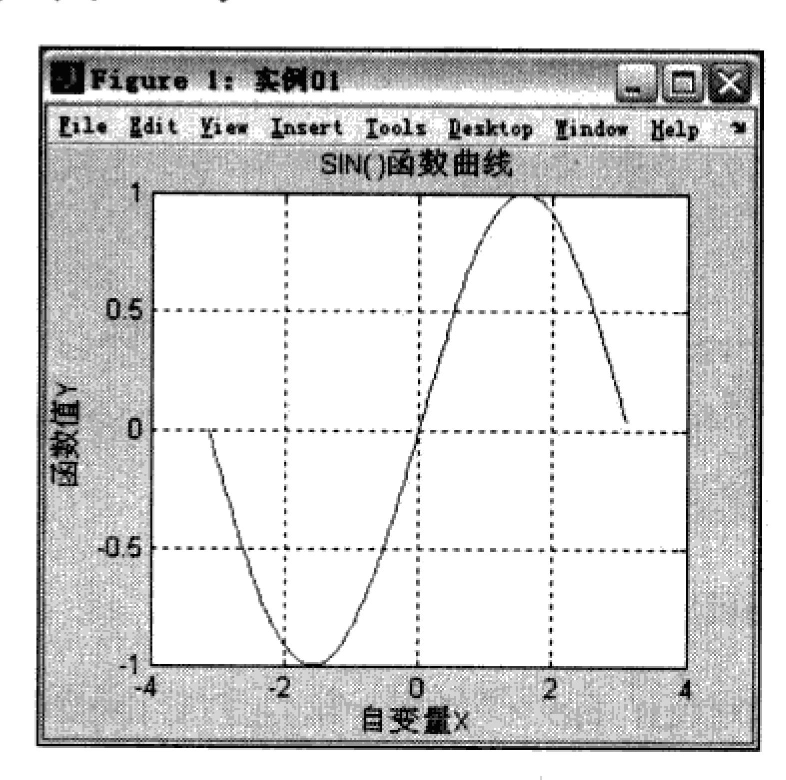


图 7.7　正弦曲线绘制

【实例讲解】 用户可以在 MATLAB 中直接输入上面的代码，查看程序的结果。

7.2.11 break 命令——结束循环

【语法说明】 break；

【功能介绍】 与循环结构相关的语句还有 break 语句，用于终止循环的执行。当在循环体内执行到该语句时，程序将跳出循环，继续执行循环语句的下一语句。

【实例 7.28】 在 MATLAB 中寻找 Fibonaci 数列中第一个大于 600 的元素及其数组标号。

```
function Fib
n=50;
a=ones(1,n);
for i=3:n
    a(i)=a(i-1)+a(i-2);
    if a(i)>=600
        a(i)
        break;
    end
end
```

得到的结果为：

```
ans =

   610


i =

    15
```

【实例讲解】 break 命令结束了整个的循环，所以，当遇到第一个大于 600 的数值（也就是 610）时，程序执行到 break 命令后跳出，不再向下执行，元素的位置标号是 15。

7.2.12 return 命令——正常退出

【功能介绍】 return 命令可使正常运行的函数正常退出，返回调用它的函数继续运行，经常用于函数的末尾用来正常结束函数的运行。

【实例7.29】 在程序中使用return命令。

```
fuction d=det(A)

If isempty(A)
D=1;
return
else
D=3
End
```

【实例讲解】 在上面的程序中，首先通过函数语句来判断参数A的类型，当A是空数组时，直接返回D=1，然后结束程序代码，否则D=3。

7.2.13 keyboard命令——停止文件执行并转交控制

【语法说明】 keyboard;

【功能介绍】 将该命令放到M文件中，将停止文件的执行并将控制权交给键盘。通过提示符K来显示一种特殊状态，只有当用户使用return命令结束输入后，控制权才交换程序。在M文件中使用该命令，对程序的调试和在程序运行中修改变量都十分方便。

【实例7.30】 在程序中使用keyboard命令。

在MATLAB的命令窗口中输入如下的内容：

```
>> keyboard
K>> for i=1:10
if i==6
continue
end
if i==5
break
end
end
K>>
K>> return
>>
```

【实例讲解】 从上面的程序可以看出，当输入keyboard命令后，在提示符的前面会显示K提示符号，而当用户输入return命令后，

提示符恢复正常的提示效果。

7.2.14 error 和 warning 命令

【语法说明】

- error('message')：显示出错信息 message，终止程序执行。
- error('errorstring'，'dlgname')：显示出错信息的对话框，对话框的标题为 dlgname。
- warning('message')：显示警告信息 message，程序继续执行。

【功能介绍】 显示错误信息或警告信息，第一个命令显示信息后终止程序的执行，第二个命令显示信息后不终止程序的执行，程序继续。

【实例 7.31】 建立 M 文件，演示出现对话框的命令格式。

```
sum1=0;
sum2=0;
n=input('Enter the number of points:');
if n<2
   errordlg('Not enough input data','Data error');
else
   errordlg('This OK','Data message');
end
```

得到的结果如图 7.8 所示。

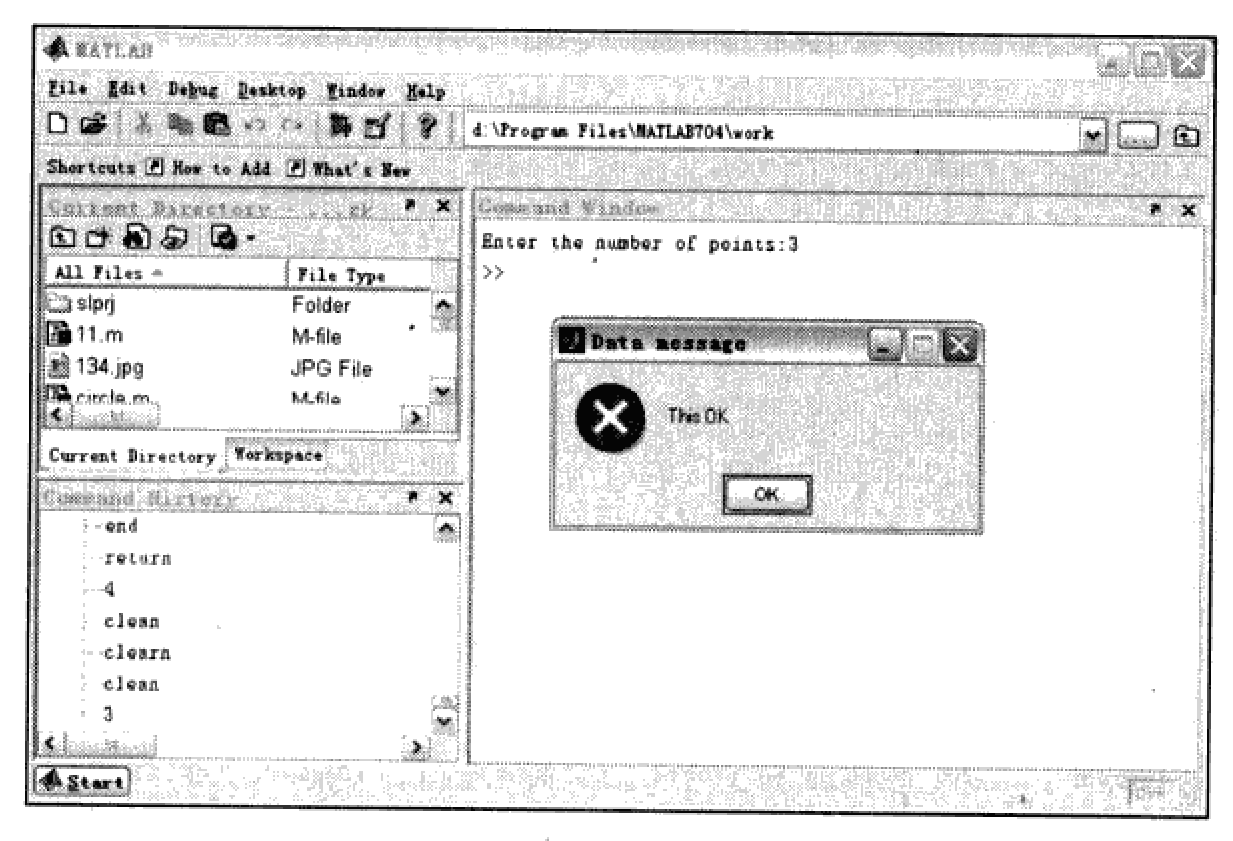


图 7.8 error 窗口演示 1

否则得到的结果如图 7.9 所示。

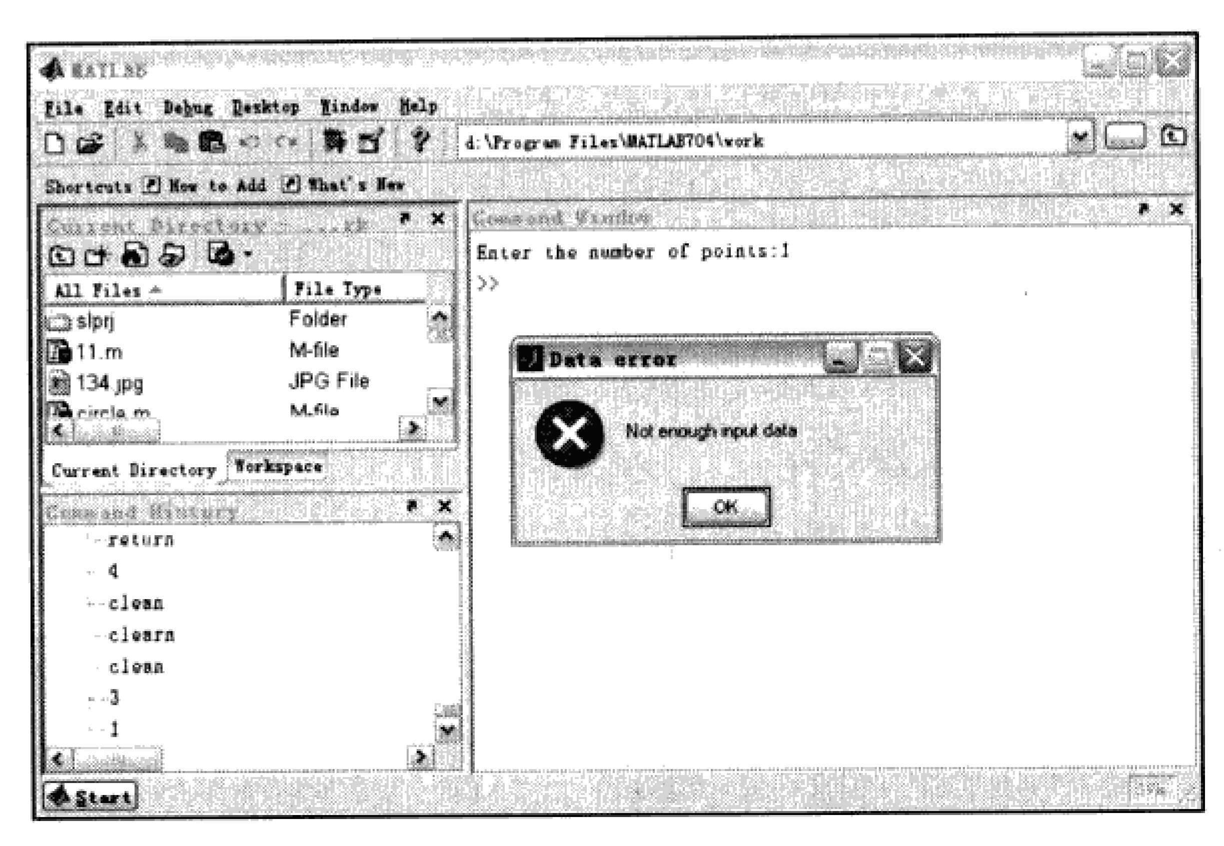


图 7.9 error 窗口演示 2

【实例讲解】 通过上面的结果可以看出在 MATLAB 的命令行中输入的数值为 3，这个数值大于 2，所以提示窗口为“This OK”。

【实例 7.32】 修改上面的程序代码，并运行程序。

（1）打开刚刚建立的 M 文件，并修改代码如下：

```
sum1=0;
sum2=0;
n=input('Enter the number of points:');
if n<2
    error('Not enough input data');
else
    error('This OK');
end
```

（2）在 MATLAB 命令行中运行该 M 文件，输入数值 1 得到的结果如下所示：

```
Enter the number of points:1
??? Error using ==> Untitled3
Not enough input data

Error in ==> Untitled3 at 5
    error('Not enough input data');
```

（3）如果输入数值 6，得到的结果如下所示：

```
Enter the number of points:6
??? Error using ==> Untitled3
This OK

Error in ==> Untitled3 at 7
    error('This OK');
```

【实例讲解】 上面的程序中，输入数值 6 同样显示错误信息，在后面的程序将进行修改。

【实例 7.33】 把上面的程序作一下小小的修改，将 error 命令改为 warning 命令，程序代码如下：

```
sum1=0;
sum2=0;
n=input('Enter the number of points:');
if n<2
   warning('Not enough input data');
else
   warning('This OK');
end
```

（1）输入数值 1，得到的结果如下：

```
Enter the number of points:1
Warning: Not enough input data
> In Untitled3 at 5
>>
```

（2）输入数值 6，得到的结果如下：

```
Enter the number of points:6
Warning: This OK
> In Untitled3 at 7
>>
```

【实例讲解】 上面的结果中检测出程序代码的各种结果。

7.2.15 循环的嵌套

【语法说明】 如果一个循环结构的循环体又包括一个循环结构，就称为循环的嵌套，或称为多重循环结构。

【实例 7.34】 若一个数等于它的各个真因子之和，则称该数为完数，如 6=1+2+3，所以 6 是完数。求[1,500]之间的全部完数。

```
for m=1:500
s=0;
for k=1:m/2
if rem(m,k)==0
s=s+k;
end
end
if m==s
   disp(m);
end
end
```

【实例讲解】 通过循环嵌套求出 1~500 之间的全部完数。

7.3 函数文件和脚本文件

函数文件是 MATLAB 中非常灵活，非常适用的一种工具。它有自己固定的格式和储存方式及函数的调用形式和参数传递模式，下面将对函数文件的相关知识做一简要介绍。

7.3.1 M 脚本文件

对于一些比较简单的问题，从指令窗口中直接输入指令进行计算是十分简单的事。但随着指令数的增加，或随控制流复杂度的增加，或重复计算要求的提出，直接从指令窗进行计算就显得有些繁琐。而此时使用脚本文件最为适宜。“脚本”本身反映这样一个事实：MATLAB 只是按文件所写的指令执行。

这种文件的构成比较简单。其特点是：

- 它只是一串按用户意图排列而成的（包括控制流向指令在内的）MATLAB 指令集合；
- 脚本文件运行后，产生的所有变量都驻留在 MATLAB 基本工作空间（Base workspace）中。只要用户不使用 clear 指令加以清除，只要 MATLAB 指令窗不关闭，这些变量将一直保存在基本工

作空间中。基本空间随 MATLAB 的启动而产生；只有当关闭 MATLAB 时，该基本空间才被删除。

7.3.2 函数文件的基本结构

【语法说明】 函数文件由 function 语句引导，其基本结构为：

```
function 输出形参表=函数名(输入形参表)
%      注释说明部分
函数体语句
```

其中以 function 开头的一行为引导行，表示该 M 文件是一个函数文件。函数名的命名规则与变量名相同。输入形参为函数的输入参数，输出形参为函数的输出参数。当输出形参多于一个时，则应该用方括号括起来。

【功能介绍】 相当于程序中的函数，在别的 MATLAB 程序中可以直接调用函数文件中的函数。

【实例 7.35】 编写函数文件求半径为 r 的圆的面积和周长。

```
function [s,p]=fcircle(r)
%CIRCLE  calculate the area and perimeter of a circle of radii r
%r          圆半径
%s          圆面积
%p          圆周长
s=pi*r*r;
p=2*pi*r;
```

【实例讲解】 这样就编写了一个求圆的面积和周长的函数。

7.3.3 函数调用

【语法说明】 函数调用的一般格式是：

```
[输出实参表]=函数名(输入实参表)
```

要注意的是，函数调用时各实参出现的顺序、个数，应与函数定义时形参的顺序、个数一致，否则会出错。函数调用时，先将实参传递给相应的形参，从而实现参数传递，然后再执行函数的功能。

【功能介绍】 有利于 M 文件的编写，缩短开发速度。

【实例 7.36】 利用函数文件，实现直角坐标(x,y)与极坐标(ρ，θ)之间的转换。

```
    函数文件 tran.m:
function [rho,theta]=tran(x,y)
rho=sqrt(x*x+y*y);
theta=atan(y/x);
    调用 tran.m 的命令文件 main1.m:
x=input('Please input x=:');
y=input('Please input y=:');
[rho,the]=tran(x,y);
rho
the
```

【实例讲解】 在 MATLAB 中，函数可以嵌套调用，即一个函数可以调用别的函数，甚至调用它自身。一个函数调用它自身称为函数的递归调用。

【实例 7.37】 利用函数的递归调用，求 n!。

```
%求 n!需要求(n-1)!,这时可采用递归调用
function f=factor(n)
if n<=1
   f=1;
else
   f=factor(n-1)*n;       %递归调用求(n-1)!
end
```

【实例讲解】 用户可以在其他的地方调用 factor 函数，查看运算的结果。

7.3.4 函数参数的可调性

【语法说明】 在调用函数时，MATLAB 用两个永久变量 nargin 和 nargout 分别记录调用该函数时的输入实参和输出实参的个数。只要在函数文件中包含这两个变量，就可以准确地知道该函数文件被调用时的输入/输出参数个数，从而决定函数如何进行处理。

【实例 7.38】 nargin 用法示例。

（1）编写原始函数的 M 文件。具体的代码如下：

```
函数文件 examp.m:
function fout=charray(a,b,c)
if nargin==1
   fout=a;
elseif nargin==2
   fout=a+b;
elseif nargin==3
   fout=(a*b*c)/2;
end
```

（2）编写调用函数的代码。具体的代码如下：

```
命令文件 mydemo.m:
x=[1:3];
y=[1;2;3];
examp(x)
examp(x,y')
examp(x,y,3)
```

（3）运行上面的 M 文件，得到的结果如下：

```
>> mydemo
ans =

    1     2     3
ans =

    2     4     6
ans =

   21
```

【实例讲解】 从上面的结果中可以看出，调用函数可以得出正确的结果。

7.3.5 全局变量与局部变量

【语法说明】 全局变量用 global 命令定义，格式为：

```
global 变量名
```

【功能说明】 使定义的变量得到更广泛的应用。

【实例 7.39】 全局变量应用示例。

先建立函数文件 mm.m，该函数将输入的参数加权相加。

```
function f=mm(x,y)
global ALPHA BETA
f=ALPHA*x+BETA*y;
```

在命令窗口中输入：

```
global ALPHA BETA
ALPHA=1;
BETA=2;
s=wadd(1,2)
```

【实例讲解】 和其他的编程语言类似，MATLAB中的变量也有对应的作用域。

7.3.6 M函数文件举例

【实例7.40】 编写一个M函数文件，使其具有以下功能：

- 根据指定的半径，画出蓝色圆周线；
- 可以通过输入字符串，改变圆周线的颜色、线型；
- 假如需要输出圆面积，则绘出圆。

（1）具体的M文件代码如下：

```
function [S,L]=exm01(N,R,str)
% exm060201.m    The area and perimeter of a regular
polygon (正多边形的面积和周长)
% N The number of sides
% R The circumradius
% str  A line specification to determine line type/color
% S The area of the regular polygon
% L The perimeter of the regular polygon
% exm01(N,R,str) 用str指定的线画外接半径为 R 的正 N 边形
% S=exm01(...)   给出多边形面积 S ,并画相应正多边形填色图
% [S,L]=exm01(...)   给出多边形面积 S 和周长 L,并画相应正多
边形填色图

switch nargin
   case 0
      N=100;R=1;str='-b';
   case 1
      R=1;str='-b';
   case 2
      str='-b';
   case 3
      ;
   otherwise
```

```
      error('输入量太多。');
end;
t=0:2*pi/N:2*pi;
x=R*sin(t);y=R*cos(t);
if nargout==0
   plot(x,y,str);
elseif nargout>2
   error('输出量太多。');
else
   S=N*R*R*sin(2*pi/N)/2;
   L=2*N*R*sin(pi/N);
   fill(x,y,str)
end
axis equal square
box on
shg
```

（2）输入代码参数，查看运行的结果：

```
[S,L]=exm01(6,2,'-g') %-g 表示用 green（绿色）来填充正 6 边形
S =
   10.3923
L =
   12.0000
```

经过这个函数文件生成的图形如图 7.10 所示。

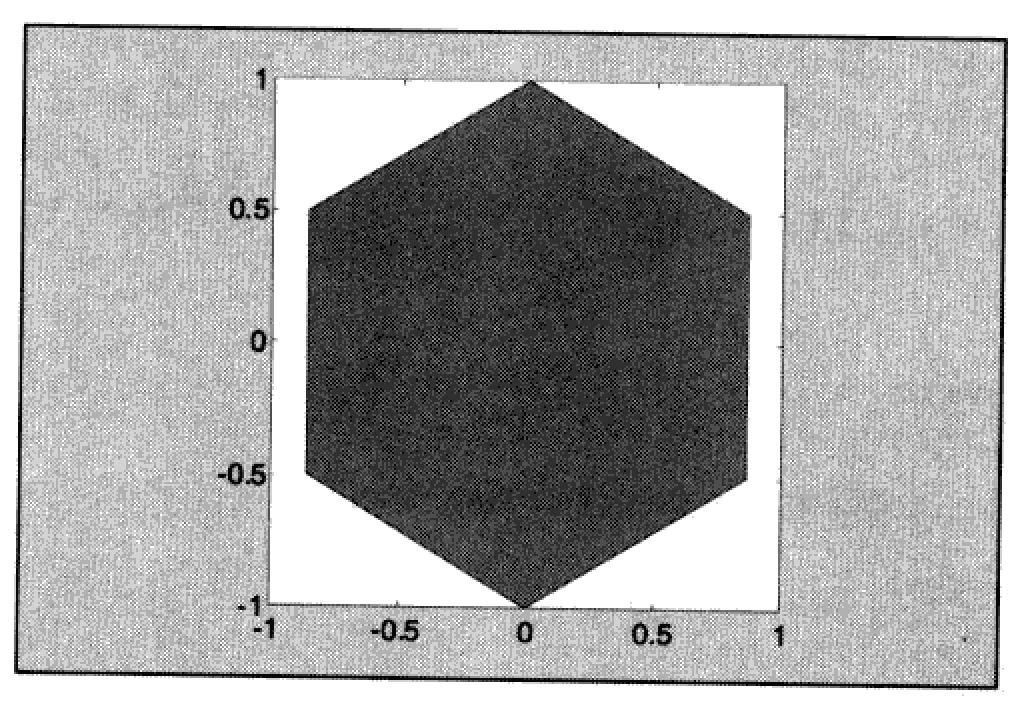


图 7.10　绿色正六边形

【实例讲解】 用户可以自行修改函数的参数，查看不同的结果。

7.4 变量的检测传递和限权使用函数

在函数中参数的传递过程中，有时候需要改变参数的长度，或者创建内联的函数。在下面的小节中主要讲解有关这两个方面的知识。通过这些内容的讲解，会极大地增加程序设计的灵活性。

7.4.1 变长度输入输出变量

【语法说明】 在 MATLAB 中 nargin 和 nargout 分别表示输入的变量个数和输出的变量个数。

【实例 7.41】 绘制圆或者圆环。

```
function varargout = ringzy(r,varargin)
%RINGZY  Plot a ring and calculate the area of the ring.
%      r 基圆半径
% 调用格式
%[x1,y1,x2,y2,s1,s2]=ringzy(r,r2,'PropertyName','Pro
pertyValue',...)
%(1) 无输出时,绘圆或环。
%(2) 有输出时,不绘图。
%        (x1,y1),(x2,y2)分别是两个圆的坐标点;
%         s1 是基圆面积;
%         s2 为正值时,表示内环面积;为负值时,表示外环面积
vin=length(varargin);Nin=vin+1;%<11>
error(nargchk(1,Nin,nargin))    %检查输入变量数目是否合适
if nargout>6%检查输出变量数目是否合适
   error('Too many output arguments')
end
t=0:pi/20:2*pi;x=r*exp(i*t);s=pi*r*r;
if nargout==0
   switch Nin
   case 1
      plot(x,'b')
   case 2
      r2=varargin{1};%<22>
```

```
        x2=r2*exp(i*t);
        plot(x,'b');hold on ;
        plot(x2,'b');hold off
     otherwise
        r2=varargin{1};%<26>
        x2=r2*exp(i*t);
        plot(x,varargin{2:end}); hold on        %利用元胞数
组设置对象属性     <28>
        plot(x2,varargin{2:end});hold off      %利用元胞数
组设置对象属性      <29>
     end;
     axis('square')
  else
     varargout{1}=real(x);varargout{2}=imag(x);   %<33>
     varargout{5}=pi*r*r;varargout{6}=[]; %<34>
     if Nin>1
        r2=varargin{1};%<36>
        x2=r2*exp(i*t);
        varargout{3}=real(x2);varargout{4}=imag(x2); %<38>
        varargout{6}=pi*(r^2-r2^2);%<39>
     end;
  end
```

得到的结果如图 7.11 所示。

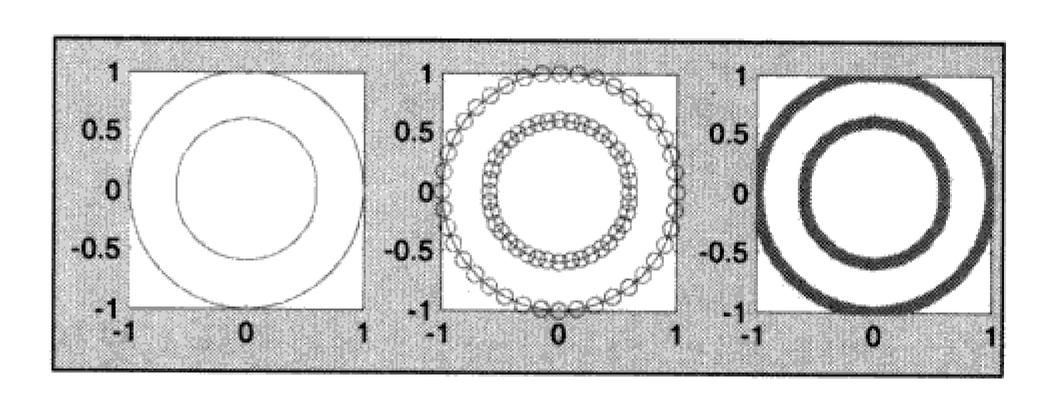


图 7.11　变长度输入变量不同调用格式产生的图形

【实例讲解】 这个例子中根据变量输出的不同情况，决定画圆环还是画圆。

7.4.2　内联函数创建

【实例 7.42】 内联函数的第一种创建格式，使内联函数适于“数组运算”。

```
clear,F1=inline('sin(rho)/rho') %第一种格式创建内联函数
F1 =
     Inline function:
     F1(rho) = sin(rho)/rho
f1=F1(2)%内联函数的一种使用方法
f1 =
0.4546
FF1=vectorize(F1)%产生适于"数组运算"的内联函数
xx=[0.5,1,1.5,2];ff1=FF1(xx)
FF1 =
     Inline function:
     FF1(rho) = sin(rho)./rho
ff1 =
     0.9589    0.8415    0.6650    0.4546
```

【实例讲解】 在上面的例子中，使用 vectorize 函数将输入变量数组化。

【实例 7.43】 演示第一种内联函数创建格式的缺陷（含向量的多变量输入的赋值）。

```
G1=inline('a*exp(x(1))*cos(x(2))'),G1(2,[-1,pi/3])
G1 =
     Inline function:
     G1(a) = a*exp(x(1))*cos(x(2))
??? Error using ==> inline/subsref
Too many inputs to inline function.
G2=inline('a*exp(x(1))*cos(x(2))','a','x'),G2(2,[-1,
pi/3])
G2 =
     Inline function:
     G2(a,x) = a*exp(x(1))*cos(x(2))
ans =
     0.3679
```

【实例讲解】 从上面的例子可以看出，MATLAB 会显示错误信息。

【实例 7.44】 产生向量输入、向量输出的内联函数。

```
Y2=inline('[x(1)^2;3*x(1)*sin(x(2))]')
argnames(Y2)                    %观察内联函数的输入总量
Y2 =
     Inline function:
     Y2(x) = [x(1)^2;3*x(1)*sin(x(2))]
```

```
ans =
'x'


x=[4,pi/6];%向量输入的赋值
y2=Y2(x)%获得向量输出
y2 =
   16.0000
    6.0000
```

【实例讲解】 从上面的过程中可以看出，向量的内联函数运算结果正常。

【实例 7.45】 演示最简练格式创建内联函数（内联函数可被 feval 指令调用）。

```
Z2=inline('P1*x*sin(x^2+P2)',2)%必须是大写字母 P
Z2 =
     Inline function:
     Z2(x,P1,P2) = P1*x*sin(x^2+P2)


z2=Z2(2,2,3)     %直接计算内联函数
fz2=feval(Z2,2,2,3)%注意: 这里,应写 Z2,不能写成 'Z2'
z2 =
   2.6279
fz2 =
2.6279
```

【实例讲解】 从上面的例子中可以看出，使用 feval 函数求解的结果和直接运算的结果相同。

7.5 程序调试

程序的编写是一个程序员最关心的问题，也是一个程序编写成功与否的关键。但是，容易被忽略的程序调试环节起着至关重要的作用，正确而熟练地使用程序调试器以及合理地使用调试命令、设置断点等都会对程序的成功运行有很大的促进作用。

7.5.1 程序调试概述

一般来说，应用程序的错误有两类，一类是语法错误，另一类是运行时的错误。语法错误包括词法或文法的错误，例如函数名的拼写错、表达式书写错等。

程序运行时的错误是指程序的运行结果有错误，这类错误也称为程序逻辑错误。

7.5.2 调试器

MATLAB 程序的调试主要由 MATLAB 运行环境本身的调试工具来完成，MATLAB 的调试工具主要有 Debug 和 Breakpoints 两个菜单项。

- Debug 菜单项：该菜单项用于程序调试，需要与 Breakpoints 菜单项配合使用。
- Breakpoints 菜单项：该菜单项共有 6 个菜单命令，前两个是用于在程序中设置和清除断点的；后 4 个是设置停止条件的，用于临时停止 M 文件的执行，并给用户一个检查局部变量的机会，相当于在 M 文件指定的行号前加入了一个 keyboard 命令。

【实例 7.46】 本例对调试器的使用进行演示，目标是：对于任意随机向量，画出鲜明标志该随机向量均值、标准差的频数直方图，如图 7.12 所示，或给出绘制这种图形的数据。

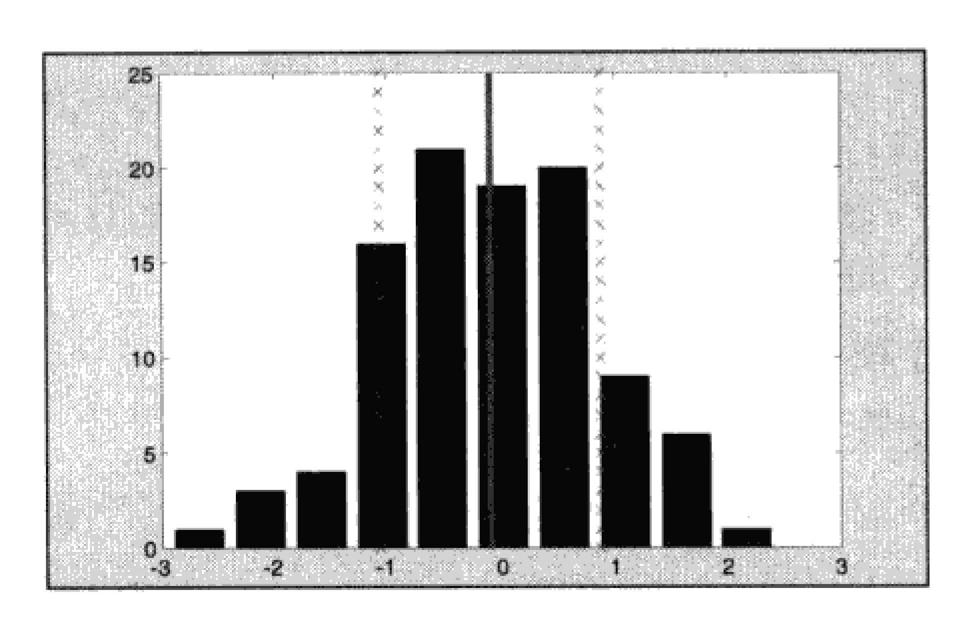


图 7.12　带均值、标准差标志的频数直方图

（1）根据题目要求写出以下两个 M 文件：

```
[histzzy.m]
function [nn,xx,xmu,xstd]=histzzy(x)
xmu=mean(x);
xstd=std(x);
[nn,xx]=hist(x);
if nargout==0
   barzzy(nn,xx,xmu,xstd)    %<6>
end


[barzzy.m]
function barzzy(nn,xx,xmu,xstd)
clf,
bar(xx,nn);hold on
Ylimit=get(gca,'YLim');
yy=0:Ylimit(2);
xxmu=xmu*size(yy);
xxL=xxmu/xmu*(xmu-xstd);
xxR=xxmu/xmu*(xmu+xstd);
plot(xxmu,yy,'r','Linewidth',3)    %<9>
plot(xxL,yy,'rx','MarkerSize',8)
plot(xxR,yy,'rx','MarkerSize',8),hold off
```

（2）初次运行以下指令后，得到运行出错的提示：

```
randn('seed',1),x=randn(1,100);histzzy(x);
??? Error using ==> plot
Vectors must be the same lengths.
Error in ==> E:\mat53\work\barzzy.m
On line 8  ==> plot(xxmu,yy,'r','Linewidth',3)
Error in ==> E:\MAT53\work\histzzy.m
On line 6  ==>   barzzy(nn,xx,xmu,xstd)
```

得到的结果如图 7.13 所示。

（3）初步分析错误原因。

（4）断点设置。

（5）在指令窗中运行以下指令，进入“动态”调试。

```
randn('seed',1),x=randn(1,100);histzzy(x);
```

得到的结果如图 7.14 所示。

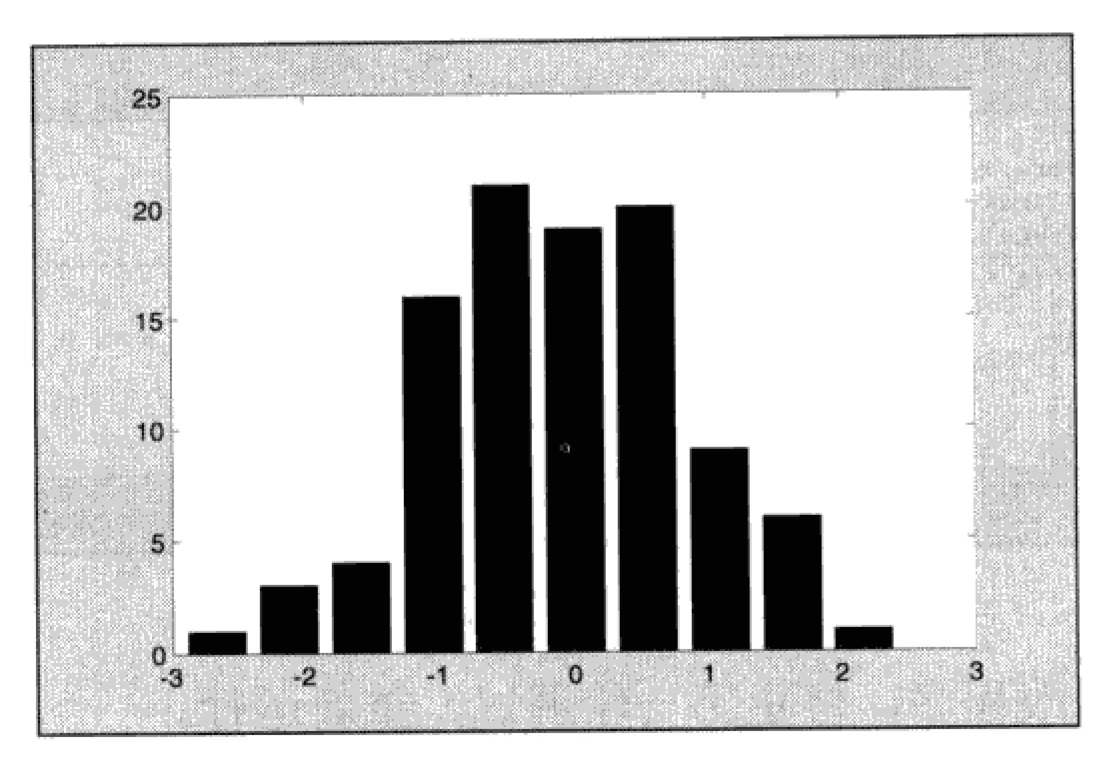


图 7.13　运行出错时所得的不完整图形

```
Stack: histzzy

function [nn,xx,xmu,xstd]=histzzy(x)
xmu=mean(x);
xstd=std(x);
[nn,xx]=hist(x);
if nargout==0
    barzzy(nn,xx,xmu,xstd)
end

histzzy.m - E...   barzzy.m - ...
Ready                                  Line 6        8:01 PM
```

图 7.14　运行暂停在断点处

（6）深入被调文件内部，如图 7.15 所示。

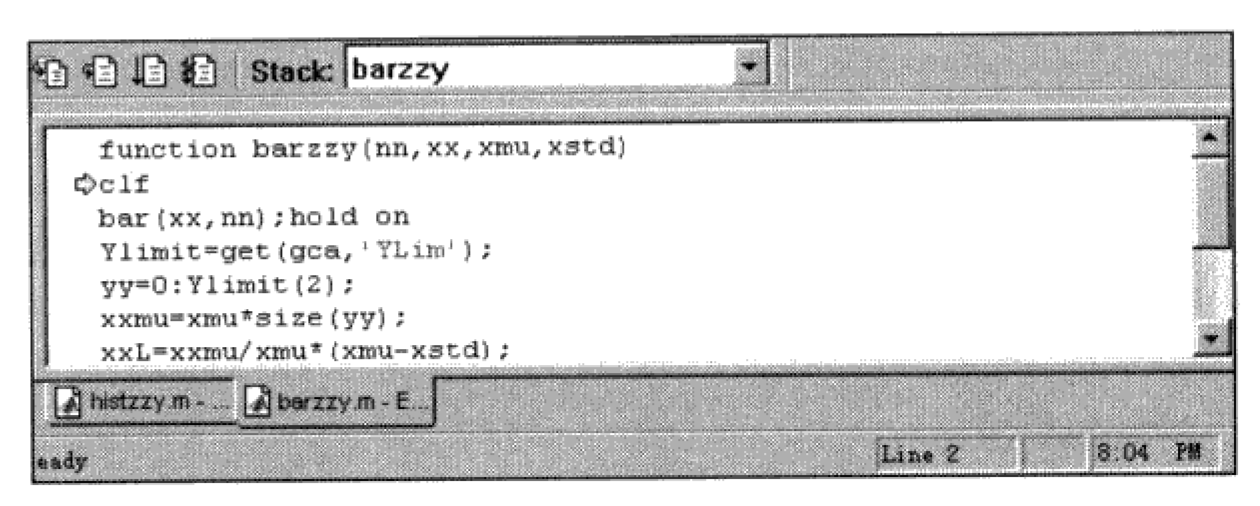


图 7.15　深入被调文件内部

（7）连续执行，直到另一个断点。如图 7.16 所示。

（8）指令窗观察法。

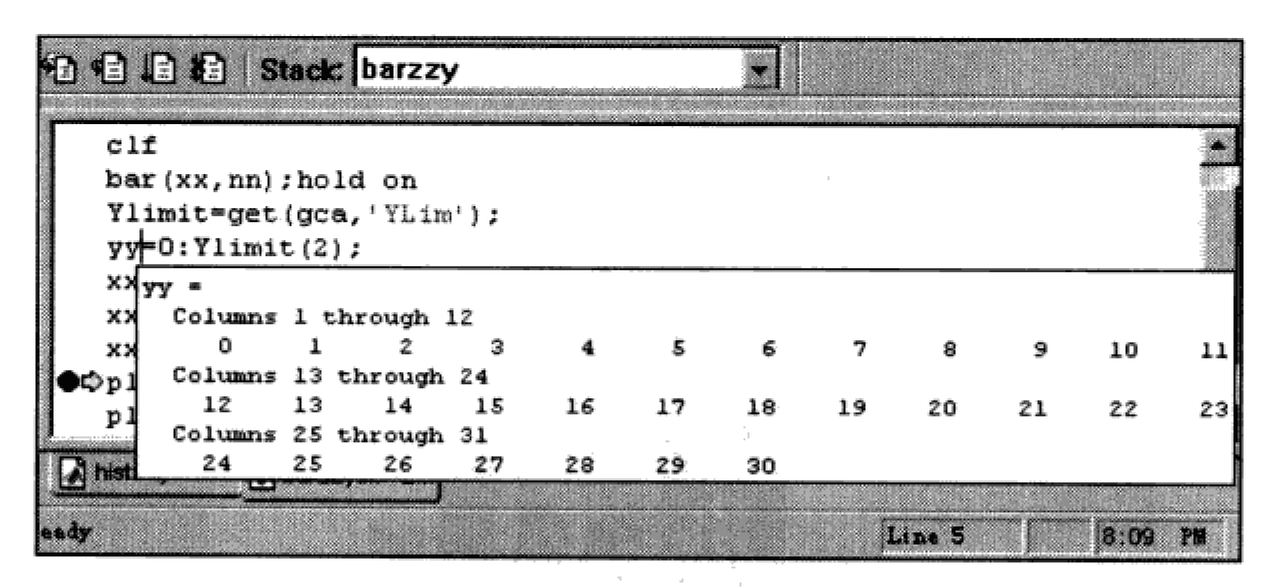


图 7.16　变量值的鼠标观察法

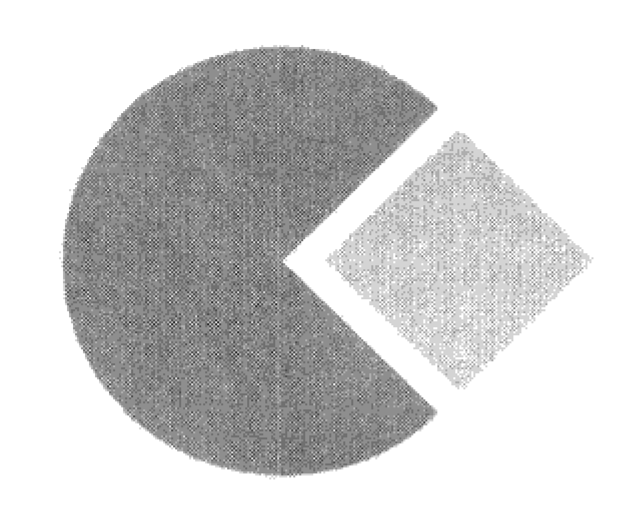
（9）修改程序，停止第一轮调试，重新运行。

```
randn('seed',1),x=randn(1,100);histzzy(x);
```

【实例讲解】 上面的例子演示了 MATLAB 中调试程序的典型过程。

7.5.3　调试命令

除了采用调试器调试程序外，MATLAB 还提供了一些命令用于程序调试。命令的功能和调试器菜单命令类似，具体使用方法请读者查询 MATLAB 帮助文档。



第8章　Simulink 命令

Simulink 是 MATLAB 的重要分支，主要用于仿真和通信领域。同时，为了提高仿真的效率和直观性，Simulink 中的任务都是可以直接通过操作实现的。所以，和 Simulink 直接相关的命令并不多。在本章中，将分类介绍各种 Simulink 常见命令。

8.1　基本命令

基本的数学函数命令是数值计算命令中很重要的部分，包括三角函数、指数函数、对数函数、排序、求绝对值、求极限和对复数的操作等。这些在数学计算中常见的运算 MATLAB 都会有相应的命令，它们为复杂的数学运算提供了方便。

8.1.1　Simulink 命令——启动模块库浏览器

【语法说明】 本命令没有参数。

【功能介绍】 通过命令行的形式调用 Simulink 模块库浏览器。

【实例 8.1】 使用命令调用 Simulink 模块库浏览器。

（1）在命令窗口中输入下面的命令：

```
>>simulink;
```

（2）启动的结果如图 8.1 所示。

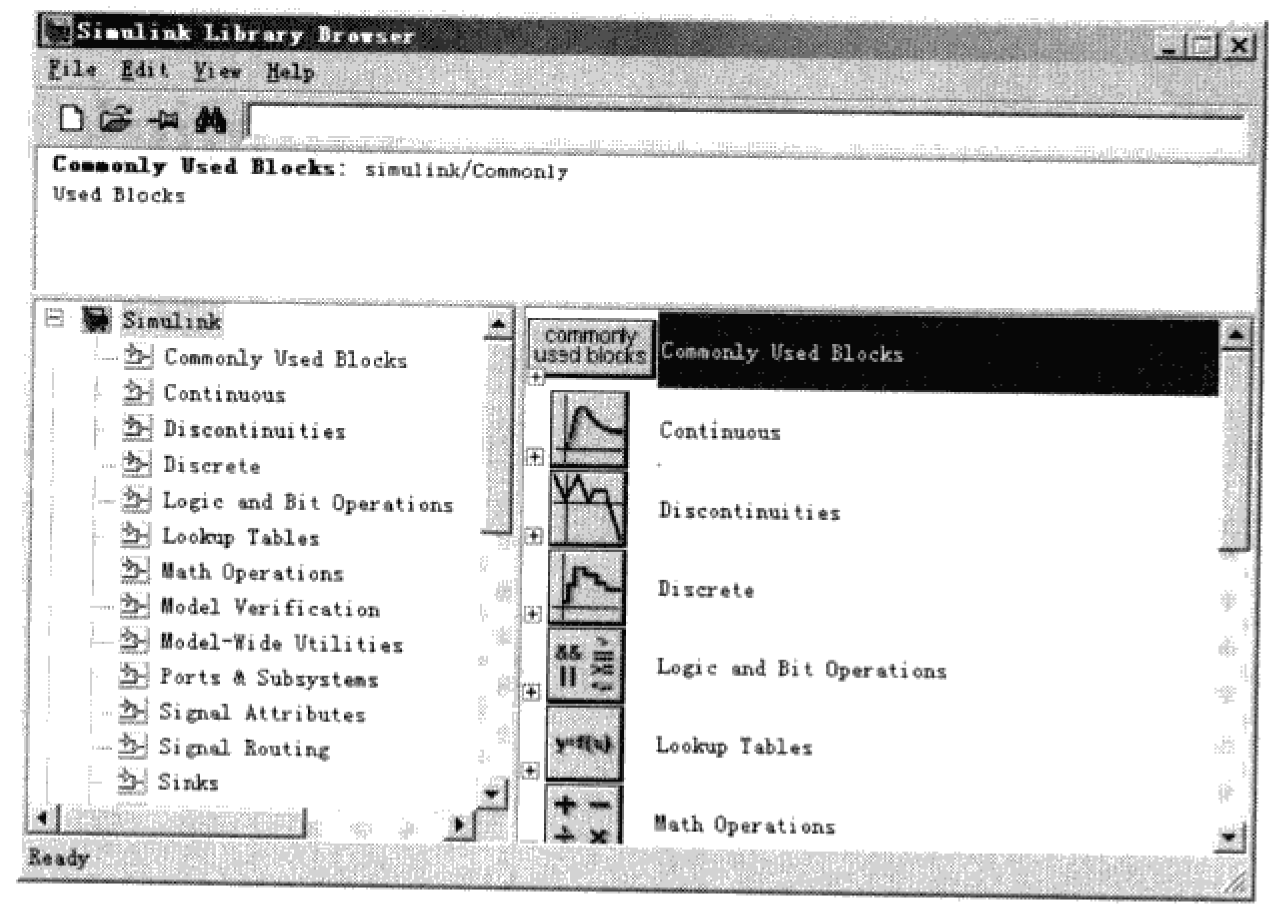


图 8.1　Simulink 模块库浏览器

【实例讲解】 在 Simulink 模块库浏览器中，用户可以选择常见的模块类型。同时，可以进行模块的常见操作。关于模块库的常用方法，请用户查看对应的帮助文件。

8.1.2　find_system 命令——查找指定的仿真系统

【语法说明】

- Systems=find_system：这个命令返回系统中所有打开的仿真系统，并将这些系统的完全路径按照次序返回到 System 中。
- Systems=find_system('parameter_name1',value1,'parameter_name2',value2,...)：按照指定的参数属性和对应的数值，查找指定的仿真系统。

【功能介绍】 按照要求查找系统中的仿真系统。

【实例 8.2】 按不同要求来查找系统中的仿真系统信息。

（1）查找系统中包含的所有的仿真系统。在命令窗口中输入下

面的命令：

```
>>find_system
```

得到的结果如下：

```
ans =

    'simulink3'
    'simulink3/Blocksets &
Toolboxes'
    'simulink3/Continuous'
    'simulink3/Continuous/Derivative'
    'simulink3/Continuous/Integrator'
    'simulink3/Continuous/Memory'
    'simulink3/Continuous/State-Space'
    'simulink3/Continuous/Transfer Fcn'
    'simulink3/Continuous/Transport
Delay'
    'simulink3/Continuous/Variable
Transport Delay'
    'simulink3/Continuous/Zero-Pole'
    'simulink3/Demos'
    'simulink3/Discrete'
    'simulink3/Discrete/Discrete
Transfer Fcn'
    'simulink3/Discrete/Discrete
Zero-Pole'
    'simulink3/Discrete/Discrete Filter'
    'simulink3/Discrete/Discrete State-Space'
    'simulink3/Discrete/Discrete-Time
Integrator'
    'simulink3/Discrete/First-Order
Hold'
    'simulink3/Discrete/Unit Delay'
    'simulink3/Discrete/Zero-Order
Hold'
    'simulink3/Functions
& Tables'
    'simulink3/Functions
& Tables/Direct Look-Up
Table (n-D)'
    'simulink3/Functions
```

```
& Tables/Fcn'
    'simulink3/Functions
& Tables/Interpolation (n-D)
using PreLook-Up'
    'simulink3/Functions
& Tables/Look-Up
Table'
MATLAB Function'
    'simulink'
    'simulink/Additional Math
& Discrete'
    'simulink/Additional Math
..........................................................................//本处省略多个结果
S-Function'
    'simulink/User-Defined
Functions/MATLAB Fcn'
    'simulink/User-Defined
Functions/S-Function'
    'simulink/User-Defined
Functions/S-Function Builder'
    'simulink/User-Defined
Functions/S-Function Examples'x))
```

（2）查找具有程序框图（Block Diagram）的仿真系统。在命令窗口输入下面的命令：

```
>> find_system('type','block_diagram')
```

得到的结果如下：

```
ans =

    'simulink3'
    'f14'
    'eml_lib'
    'simulink'
```

【实例讲解】 在上面的程序结果中，显示的主要是系统自带的演示仿真系统信息。通过这个命令，用户可以了解到系统自带仿真系统的详细信息。

8.1.3 load_system 命令——加载指定的仿真系统

【语法说明】 load_system('sys')：参数 sys 表示指定记载的仿真

系统名称。

【功能介绍】 这个命令的功能是加载用户指定的仿真系统，但是并不打开对应的系统。

【实例 8.3】 加载系统自带的仿真系统。

（1）加载系统自带的 f14 仿真系统。在命令窗口输入下面的命令：

```
>> load_system('f14');
```

系统已经加载 f14 仿真系统。用户可以在后面的步骤中引用仿真系统的参数和信息。

（2）直接在命令窗口中输入仿真系统的名称。查看 f14 系统的情况：

```
>> f14
```

在输入上面的命令后，按下 Enter 键，得到 f14 仿真系统的结果如图 8.2 和图 8.3 所示。

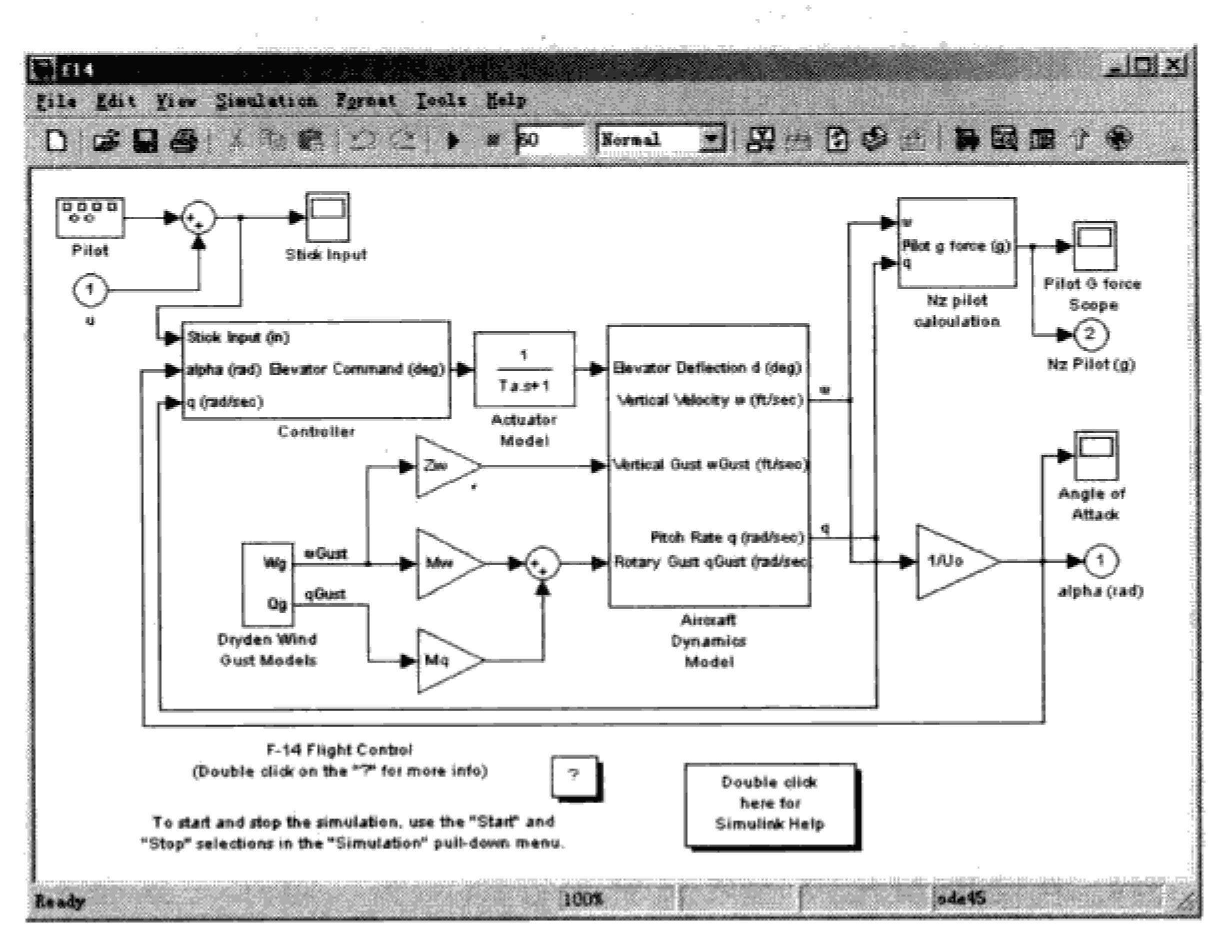


图 8.2 仿真系统框架图

【实例讲解】 从上面的结果中可以看出，通过 load_system 命令只是加载仿真系统，并没有打开仿真系统。如果用户希望直接打开仿真系统，可以直接在命令窗口中输入仿真系统的名字。

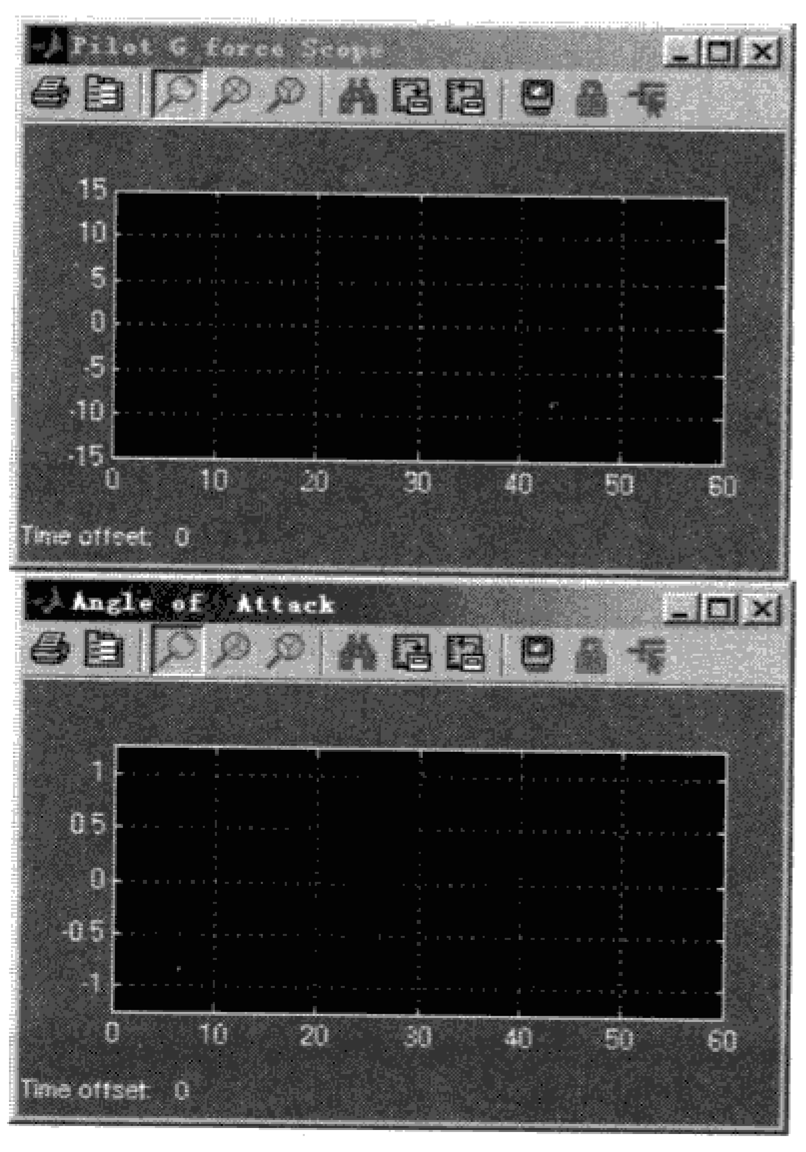


图 8.3　仿真示意图

8.1.4　open_system 命令——打开仿真系统或者子系统

【语法说明】

- open_system('sys')：打开指定的仿真系统或者子系统。
- open_system('blk')：打开指令模块相关的框架演示图，其中，参数 blk 必须是全路径参数。

> 提示　在 open_system 命令中，用户也可以为指定系统设置详细的参数说明。具体的指定方法这里就不详细说明，请用户查看对应的帮助文件。

【功能说明】 按照指定的要求，打开仿真系统或者子系统。

【实例 8.4】 设定不同的参数，打开系统自带的仿真系统

（1）打开系统自带的仿真系统 aero_radmod。在命令行中输入

下面的命令：

```
>> open_system('aero_radmod')
```

按下 Enter 键，得到的结果如图 8.4 所示。

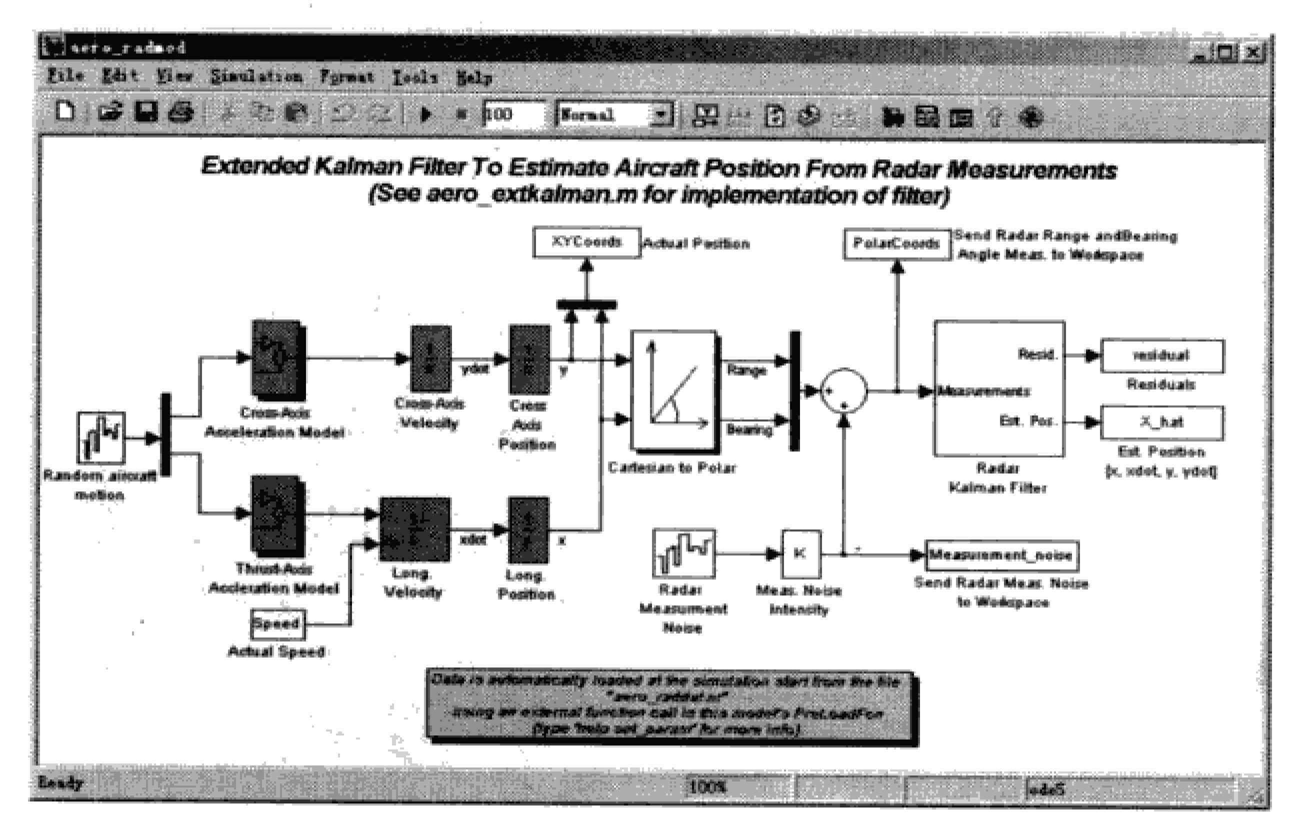


图 8.4 打开仿真系统

（2）打开这个系统中的“Radar Kalman Filter”子系统。在命令行中输入下面的命令：

```
>> open_system('aero_radmod/Radar Kalman Filter')
```

按下 Enter 键，得到的结果如图 8.5 所示。

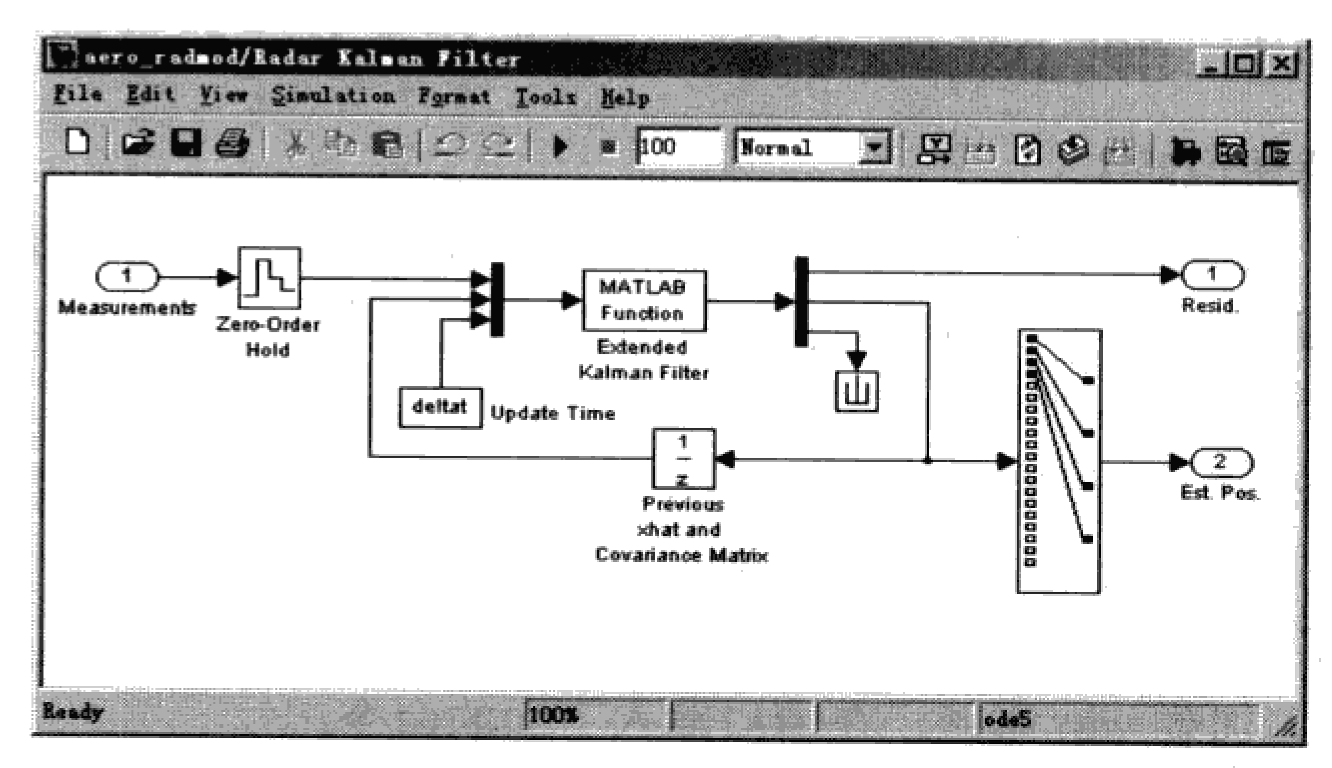


图 8.5 打开对应的子系统

（3）同时打开两个仿真系统，并将两个仿真系统的窗口垂直叠放。在命令窗口中输入下面的命令：

```
>>open_system( { 'aero_radmod','vdp' });
```

按下 Enter 键，得到的结果如图 8.6 所示。

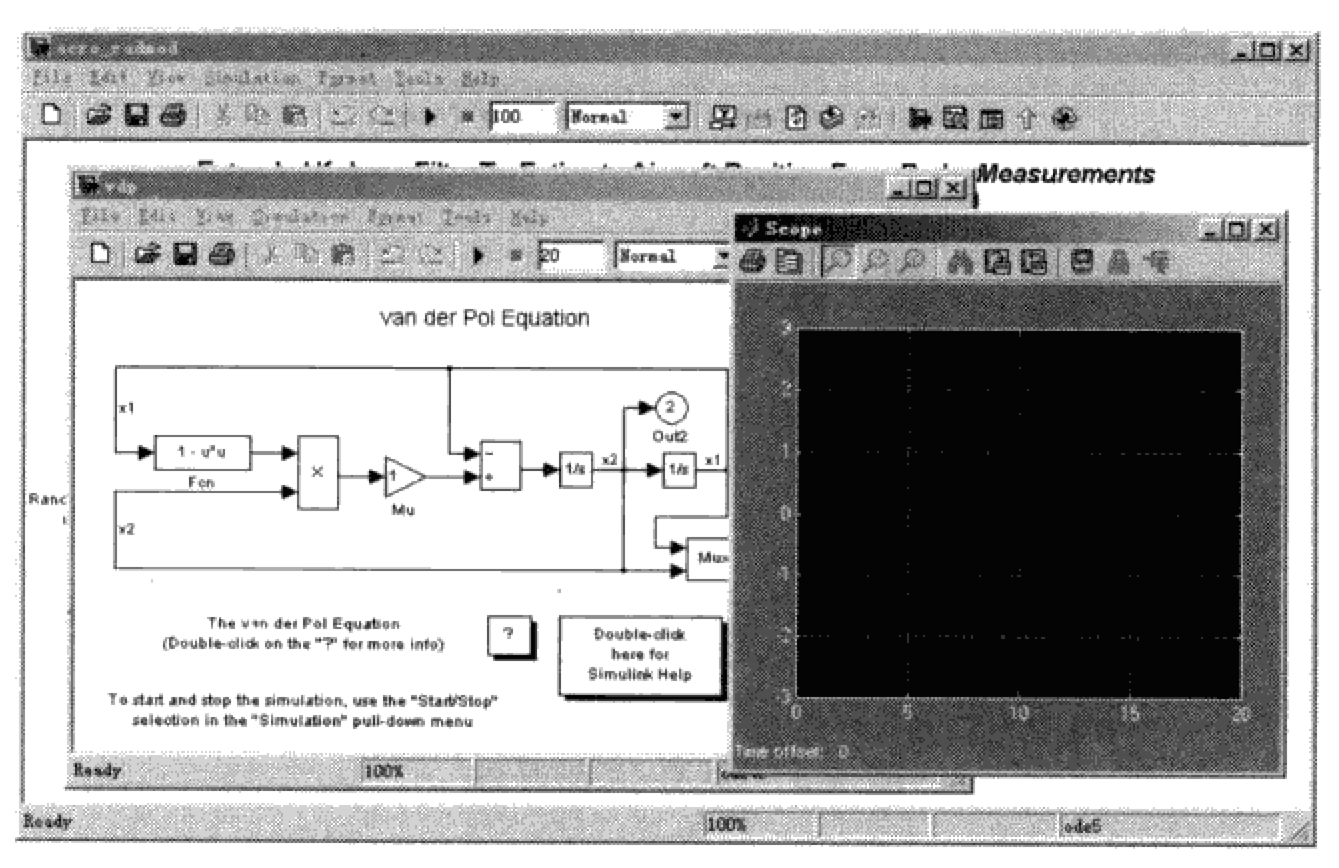


图 8.6　打开交叉的仿真系统

【实例讲解】 从上面的例子中可以看出，用户只要知道仿真系统的名字就可以用各种形式打开系统。

8.1.5　get_param 命令——获取仿真系统的参数

【语法说明】

- get_param('obj','parameter')：参数“obj”表示仿真系统或者模块的路径，参数“parameter”表示需要获取的仿真系统的参数。
- get_param(obj, 'objectparameters')：返回一个结构体，该结构体中的数值分别是这个仿真系统的参数信息。

【功能介绍】 获取仿真系统的参数。

【实例 8.5】 获取系统自带仿真系统“vdp”的参数信息。

（1）加载 vdp 系统。在窗口输入下面的命令：

```
>>load_system('vdp')
```

（2）获取 vdp 系统的 Solver 参数。在窗口输入下面的命令：

```
>> get_param('vdp','Solver')
```

得到的结果如下.

```
ans =

ode15s
```

（3）获取 vdp 系统的 StopTime 参数。在窗口输入下面的命令：

```
>>get_param('vdp','StopTime')
```

得到的结果如下：

```
ans =

3000
```

【实例讲解】 vdp 是系统自带的一个 demo 仿真系统，在窗口中输入下面的命令：

```
>> vdp
```

其对应的仿真结构图如图 8.7 所示。

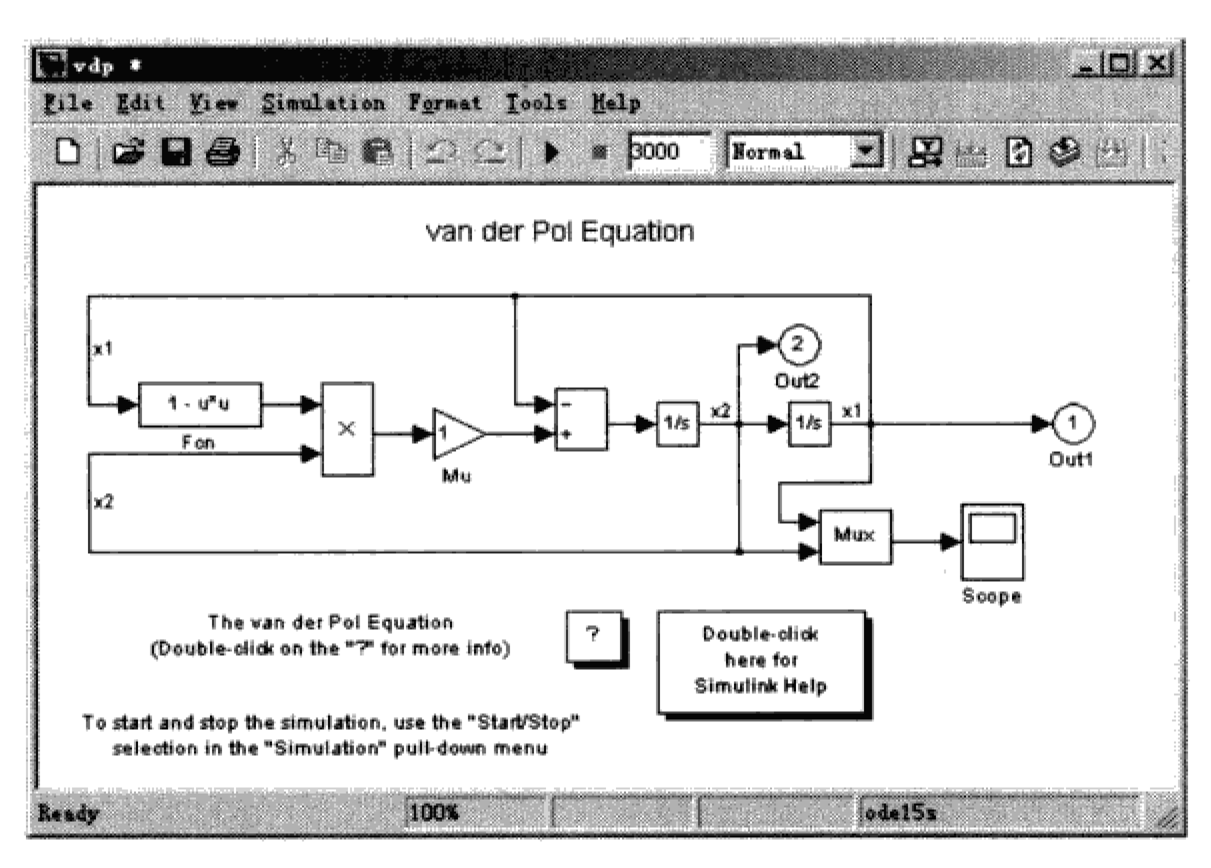


图 8.7 vdp 系统结构图

在 Simulink 中，用户可以通过仿真参数窗口直接查看对应的参数信息。具体步骤如下。

（1）在打开的仿真系统中，选择“simulation” | “Configuration

Parameters”选项，如图 8.8 所示。

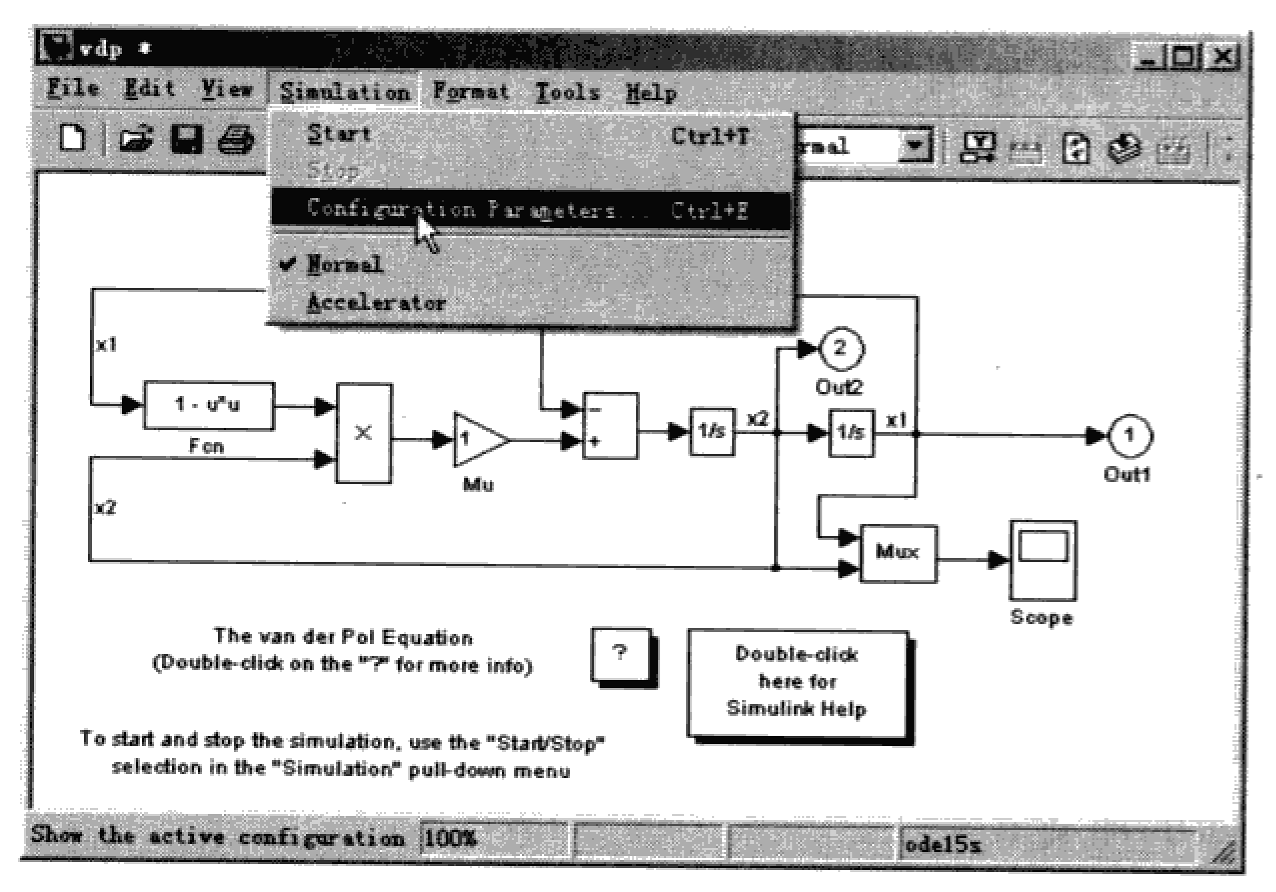


图 8.8　选择仿真参数选项

（2）选择“Solver”选项，查看“Stop time”和“Solver”参数的数值，如图 8.9 所示。

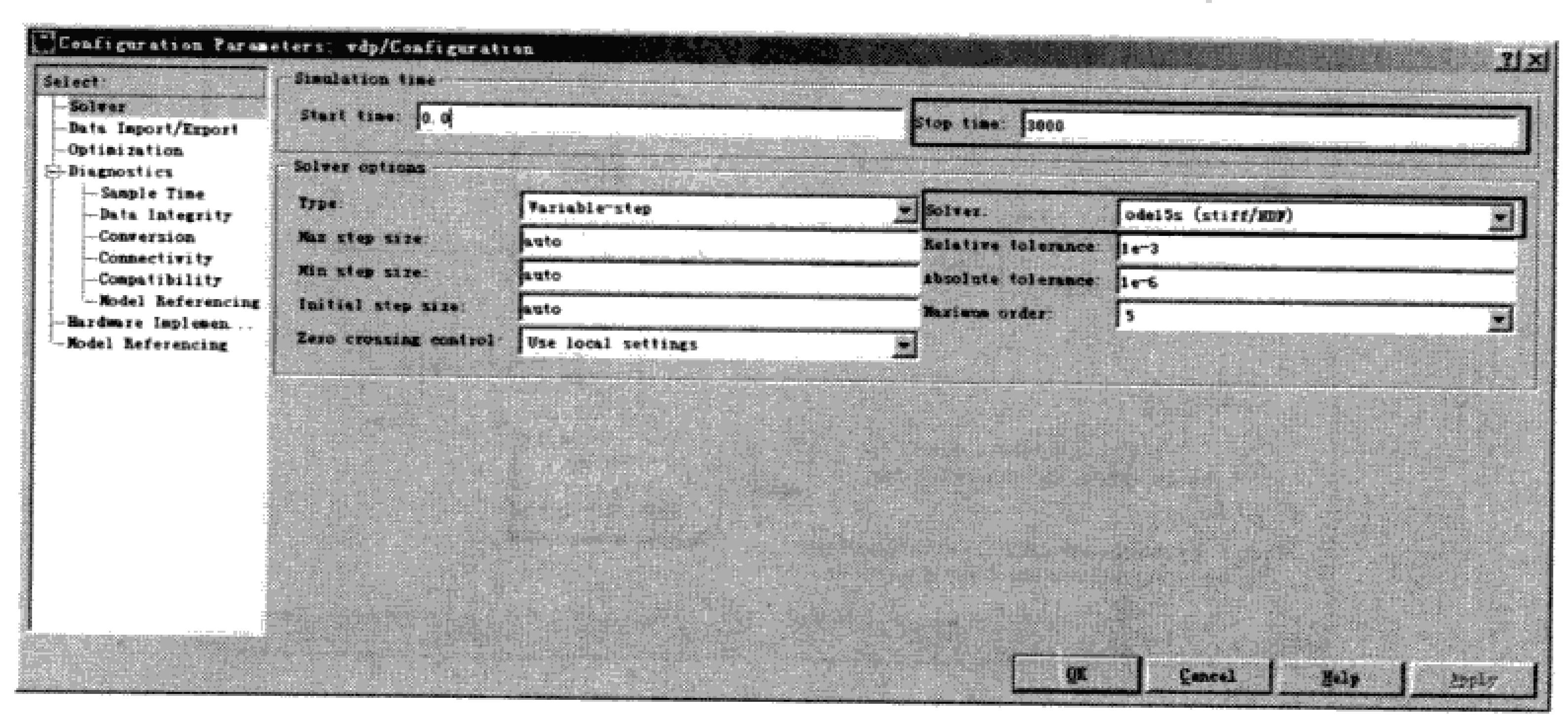


图 8.9　查看对应的参数信息

说明　从上面的表格中可以看出，使用 get_param 参数可以直接获取仿真系统对应的参数信息。

8.1.6 set_param 命令——设置仿真系统的参数

【语法说明】

set_param('obj','parameter1',value1,'parameter2',value2,...)：参数“obj”是仿真系统或者模块的路径名称，parameter 表示参数名称，value 表示参数对应的数值。

【功能介绍】 设置或者修改仿真系统的参数。

【实例 8.6】 设置或者修改仿真系统 vdp 模块的参数。

（1）加载 vdp 系统，并获取相关模块的参数信息。在命令窗口输入下面的命令：

```
>> load_system('vdp');
>> vg=get_param('vdp/Mu','Gain');
>> vfcn=get_param('vdp/Fcn','Position');
```

（2）查看模块的参数信息，结果如下：

```
>> vg
vg =
1
>> vfcn
vfcn =
    50    99   110   121
```

（3）修改对应的参数信息，结果如下：

```
>> set_param('vdp/Mu','Gain','100');
>>set_param('vdp/Fcn','Position',[49      100      109
122]);
```

在上面的命令中，用户设置了对应参数的数值。

（4）查看“Mu”模块的参数信息。在 Simulink 窗口中，选择“Edit”|“Gain Parameters”选项，如图 8.10 所示。

（5）查看 Gain 参数的数值。在打开的对话框中，查看系统的 Gain 参数的数值，如图 8.11 所示。

【实例讲解】 从上面的结果中可以看出，通过 set_param 命令，用户可以直接修改仿真系统的相关参数。

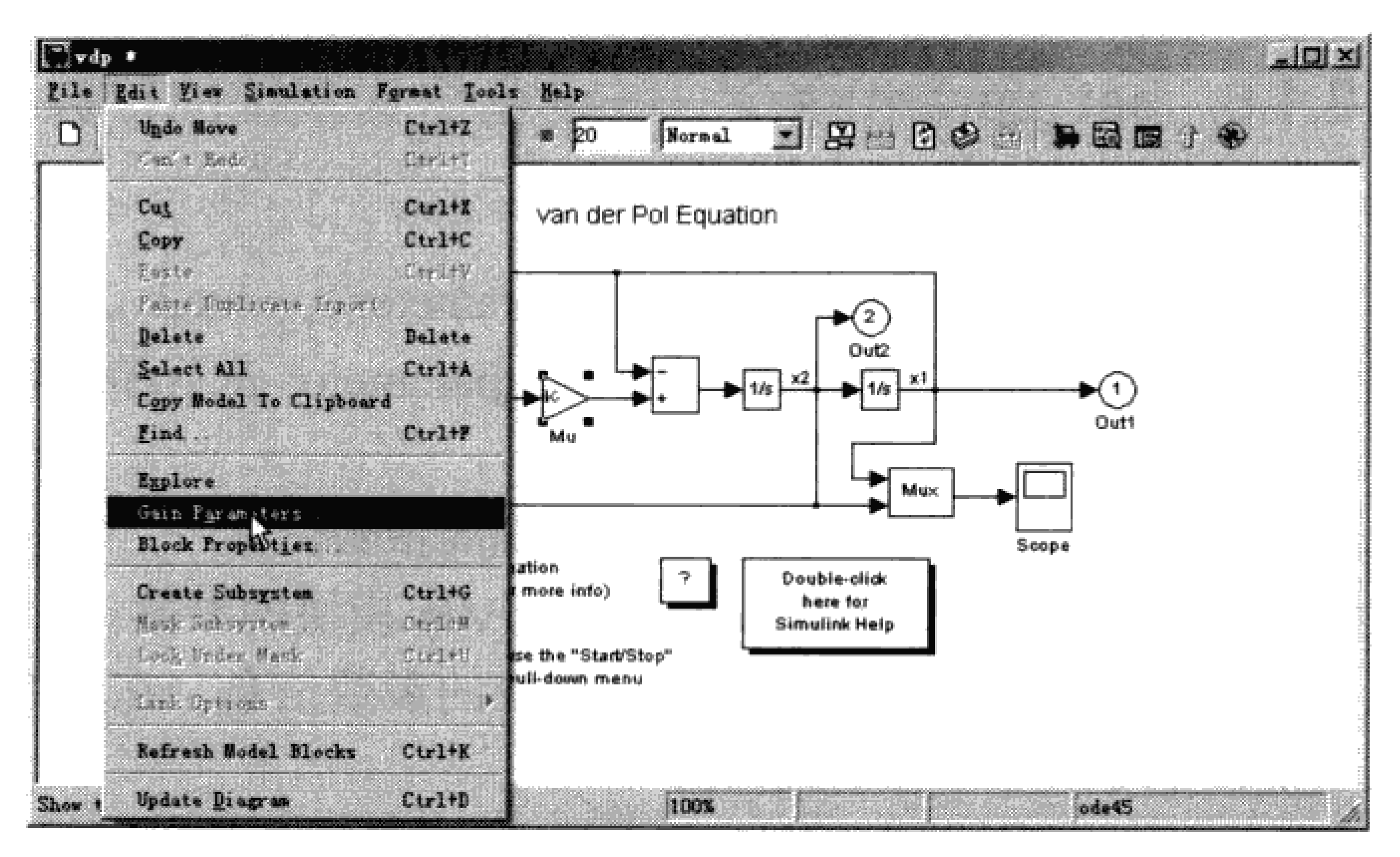


图 8.10　查看对应的参数选项

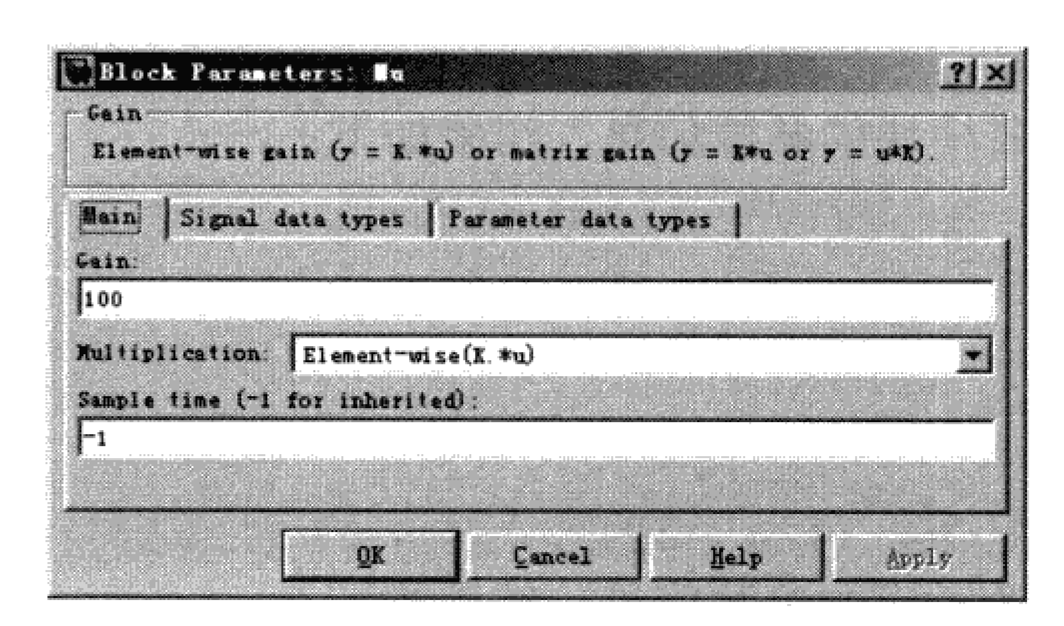


图 8.11　查看 Gain 参数的数值

8.1.7　gcs 和 gab 命令——获取当前仿真系统或模块的名称

【语法说明】

- gcs：获取当前运行的仿真系统名称。gcs 其实是 get current system 的缩写。
- gcb：获取当前仿真系统的模块名称。gab 其实是 get current block 的缩写。

【功能介绍】 获取当前系统或模块的名称。

【实例 8.7】 使用两个命令，获取系统的命令信息。

（1）打开仿真系统 vdp，然后获取系统或者模块的名称。在命令窗口输入下面的命令：

```
>> load_system('vdp');
>> sname=gcs;
>> bname=gcb;
```

（2）查看系统或者模块的名称信息，结果如下：

```
>> sname
sname =
vdp
>> bname
bname =
vdp/Out2
```

（3）关闭仿真系统，并打开新的仿真系统 f14。重新获取相应的信息。在命令窗口输入下面的命令：

```
>> close_system('vdp');
>> load_system('f14');
>> sname=gcs;
>>bname=gcb;
```

（4）查看系统或者模块的名称信息，结果如下：

```
>> sname
sname =
f14
>> bname
bname =
f14/Nz Pilot (g)
```

【实例讲解】 在上面的例子中，关闭原来仿真系统的命令很关键。如果不关闭原来的系统，直接打开新的系统，两个命令会得到相同的结果。

8.1.8 gcbh 和 getfullname 命令——获取系统的句柄和名称

【语法说明】

- gcbh：获取当前仿真系统的句柄对象。
- name=getfullname(handle)：返回由句柄所指定的系统或者模块的名称。

【功能介绍】 获取系统句柄或者名称。

【实例 8.8】 加载仿真系统，并获取对应的句柄和名称。

在命令窗口中输入下面的命令：

```
>> load_system('f14');
>> handel1=gcbh;
>> getfullname(handel1)
```

得到的结果如下：

```
ans =

f14/Nz Pilot (g)
```

【实例讲解】 这两个命令在使用 MATLAB 中会有重要的用途。

8.1.9 bdclose 命令——关闭正在打开的仿真系统窗口

【语法说明】 bdclose 命令的功能是关闭正在打开的仿真系统窗口。该命令没有参数。

【功能介绍】 关闭仿真系统窗口。

【实例 8.9】 使用 bdclose 命令关闭仿真系统的窗口。

（1）打开 vdp 仿真系统，在命令窗口中输入下面的命令：

```
>> vdp
```

得到的结果如图 8.12 所示。

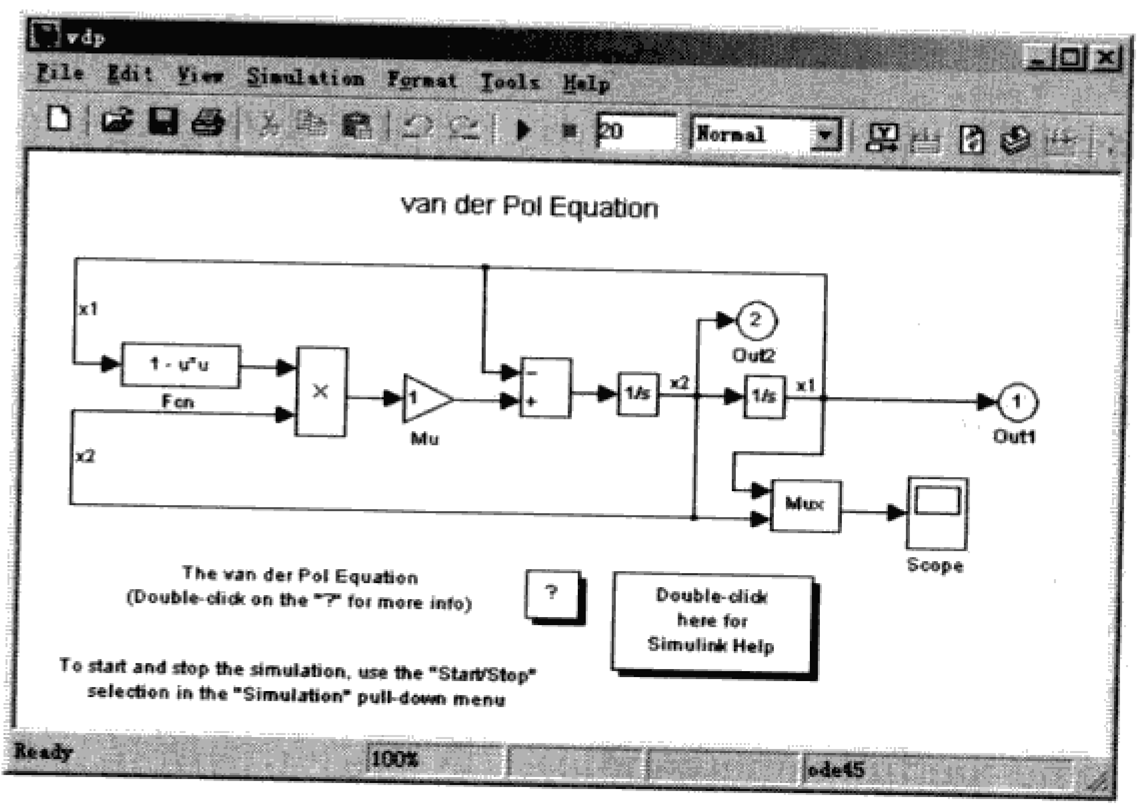


图 8.12 打开仿真系统

（2）关闭仿真系统的窗口，在命令窗口中输入下面的命令：

```
>> bdclose
```

可以发现仿真命令窗口已经关闭。

【实例讲解】 使用命令关闭窗口，可以在系统控制和开发中有广泛应用。

8.1.10 slupdate 命令——更新系统的模块

【语法说明】 slupdate(sys)：其中 sys 表示仿真系统名称。

【功能介绍】 slupdate 命令的功能是更新仿真系统的模块。

【实例 8.10】 更新仿真系统 f14。

（1）在命令窗口中输入下面的命令：

```
>> slupdate('f14')
```

（2）依次确认系统的更新信息，具体信息如下：

```
    Replace 'f14/Aircraft Dynamics Model/Vertical Velocity
w (ft//sec)'?   ([y]/n/a)
    Updating: 'f14/Aircraft Dynamics Model/Vertical
Velocity w (ft//sec)'
    Replace 'f14/Aircraft Dynamics Model/Pitch Rate q
(rad//sec)'? ([y]/n/a)
    Updating: 'f14/Aircraft Dynamics Model/Pitch Rate q
(rad//sec)'
    Replace 'f14/Controller/Elevator Command (deg)'?
([y]/n/a)
    Updating: 'f14/Controller/Elevator Command (deg)'
    Replace 'f14/Dryden Wind Gust Models/Wg'? ([y]/n/a)
    Updating: 'f14/Dryden Wind Gust Models/Wg'
    Replace 'f14/Dryden Wind Gust Models/Qg'? ([y]/n/a)
    Updating: 'f14/Dryden Wind Gust Models/Qg'
    Replace 'f14/Nz pilot calculation/Pilot g force (g)'?
([y]/n/a)
    Updating: 'f14/Nz pilot calculation/Pilot g force (g)'
    Replace 'f14/alpha (rad)'? ([y]/n/a)
    Updating: 'f14/alpha (rad)'
    Replace 'f14/Nz Pilot (g)'? ([y]/n/a)
    Updating: 'f14/Nz Pilot (g)'
```

（3）最后得到的更新信息如下：

```
    The following blocks in 'f14' were updated:
      f14/Aircraft  Dynamics  Model/Vertical  Velocity  w
(ft//sec)
      f14/Aircraft Dynamics Model/Pitch Rate q (rad//sec)
      f14/Controller/Elevator Command (deg)
      f14/Dryden Wind Gust Models/Wg
      f14/Dryden Wind Gust Models/Qg
      f14/Nz pilot calculation/Pilot g force (g)
      f14/alpha (rad)
      f14/Nz Pilot (g)
      f14/u
      f14/Aircraft  Dynamics  Model/Elevator  Deflection  d
(deg)
      f14/Aircraft  Dynamics  Model/Vertical  Gust  wGust
(ft//sec)
      f14/Aircraft  Dynamics  Model/Rotary  Gust  qGust
(rad//sec)
      f14/Controller/Stick Input (in)
      f14/Controller/alpha (rad)
      f14/Controller/q (rad//sec)
      f14/Nz pilot calculation/w
      f14/Nz pilot calculation/q
      f14/Aircraft Dynamics Model
      f14/Controller
      f14/Dryden Wind Gust Models
      f14/Dryden Wind Gust Models/Band-Limited White Noise
      f14/More Info
      f14/More Info1
      f14/Nz pilot calculation
      f14/Aircraft Dynamics Model)
```

【实例讲解】 用户可以使用其他的系统，来更新对应的模块。

8.1.11 slhelp 命令——查看 Simulink 的帮助信息

【语法说明】 这个命令没有参数。

【功能介绍】 这个命令的功能是调用 simulink 的帮助文件。

【实例 8.11】 使用 slhelp 命令调用帮助信息。

在命令窗口中输入下面的信息：

```
>> slhelp
```

得到的结果如图 8.13 所示。

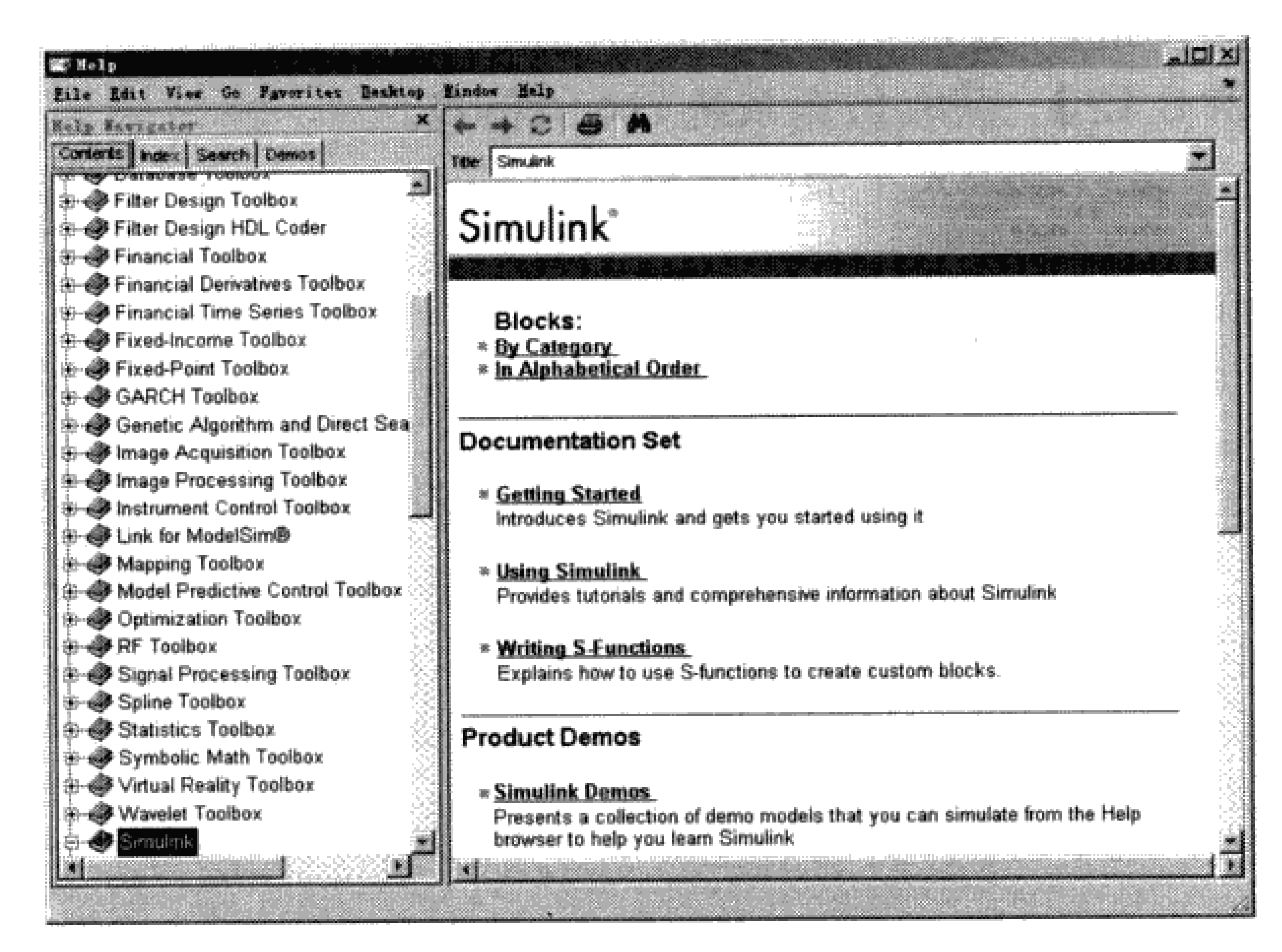


图 8.13　Simulink 的帮助信息

8.2　仿真命令

在 Simulink 中，多数的仿真结果都可以直接在仿真窗口执行和完成。但是，用户同样可以在命令窗口中使用命令设置仿真的参数，并运行仿真系统。下面详细介绍相关的仿真命令。

8.2.1　simget 命令——获取仿真系统的信息

【语法说明】

- struct = simget(model)：获取当前仿真系统的参数信息，以结构体的形式给出。
- value = simget(model,property)：返回仿真系统中指定参数的数值。
- value=simget(optionstructure,property)：返回仿真系统中指

定参数的数值。

【功能介绍】 获取仿真系统的信息。

【实例 8.12】 获取仿真系统 f14 的信息。

在命令窗口中输入下面的命令：

```
>> simget f14
```

得到的结果如下：

```
ans =

                 AbsTol: 1.0000e-006
                  Debug: 'off'
             Decimation: 1
           DstWorkspace: 'current'
         FinalStateName: ''
              FixedStep: 'auto'
           InitialState: []
            InitialStep: 'auto'
               MaxOrder: 5
             SaveFormat: 'Array'
          MaxDataPoints: 0
                MaxStep: 'auto'
                MinStep: 'auto'
           OutputPoints: 'all'
        OutputVariables: ''
                 Refine: 1
                 RelTol: 1.0000e-004
                 Solver: 'ode45'
           SrcWorkspace: 'base'
                  Trace: ''
              ZeroCross: 'on'
     ExtrapolationOrder: 4
 NumberNewtonIterations: 1
```

【实例讲解】 在上面的结果中，用户可以查看到 f14 仿真系统中所有重要的信息。

8.2.2 simset 命令——设置仿真参数

【语法说明】

- options = simset('name1',value1,'name2',value2,...)：通过设置

指定系统参数数值，设置仿真参数。

- options = simset(oldopts,'name1',value1,...)：修改原来仿真系统中的参数数值。
- options = simset(oldopts,newopts)：将原来仿真系统中的参数修改为新的仿真系统参数。

【功能介绍】 设置仿真系统的参数。

【实例 8.13】 查看当前系统中的仿真参数。

在命令窗口中输入下面的命令：

```
>> simset
```

得到的结果是：

```
              Solver: [ 'VariableStepDiscrete' |
                        'ode45' | 'ode23' | 'ode113' |
'ode15s' | 'ode23s' | 'ode23t' | 'ode23tb'|
                        'FixedStepDiscrete' |
                        'ode5' | 'ode4' | 'ode3' | 'ode2' |
'ode1' | 'ode14x' ]
              RelTol: [ positive scalar {1e-3} ]
              AbsTol: [ positive scalar {1e-6} ]
              Refine: [ positive integer {1} ]
             MaxStep: [ positive scalar {auto} ]
             MinStep: [ [positive scalar, nonnegative integer]
{auto} ]
         InitialStep: [ positive scalar {auto} ]
            MaxOrder: [ 1 | 2 | 3 | 4 | {5} ]
           FixedStep: [ positive scalar {auto} ]
  ExtrapolationOrder: [ 1 | 2 | 3 | {4} ]
NumberNewtonIterations: [ positive integer {1} ]
        OutputPoints: [ {'specified'} | 'all' ]
     OutputVariables: [ {'txy'} | 'tx' | 'ty' | 'xy' | 't'|
'x' | 'y' ]
          SaveFormat:  [   {'Array'}   |   'Structure'   |
'StructureWithTime']
       MaxDataPoints: [ non-negative integer {0} ]
          Decimation: [ positive integer {1} ]
        InitialState: [ vector {[]} ]
      FinalStateName: [ string {''} ]
               Trace: [ comma separated list of 'minstep',
```

```
'siminfo', 'compile', 'compilestats' {''}]
     SrcWorkspace: [ {'base'} | 'current' | 'parent' ]
     DstWorkspace: [ 'base' | {'current'} | 'parent' ]
        ZeroCross: [ {'on'} | 'off' ]
            Debug: [ 'on' | {'off'} ]
```

【实例讲解】 simset 命令同样可以用来设置其他仿真模型的具体参数。

8.2.3 sim 命令——运行仿真

【语法说明】

- sim('model')：使用仿真模型的参数运行整个仿真系统。
- [t,x,y]= sim('model',timespan,options,ut)：设置仿真的时间区间、输出时间记录点的向量和其他权限，然后运行仿真系统。将仿真结果返回到矩阵中。
- [t,x,y1,...,yn] = sim('model',timespan,options,ut)：参数含义和上面类似。

【功能介绍】 在命令窗口中运行仿真系统。

【实例 8.14】 创建仿真系统，该系统的目的在于求解微分方程 $\sin x-\frac{1}{2}x=x'$。然后，在命令窗口中运行仿真系统。分析结果。

（1）打开 Simulink 窗口，然后依次添加模块，结果如图 8.14 所示。

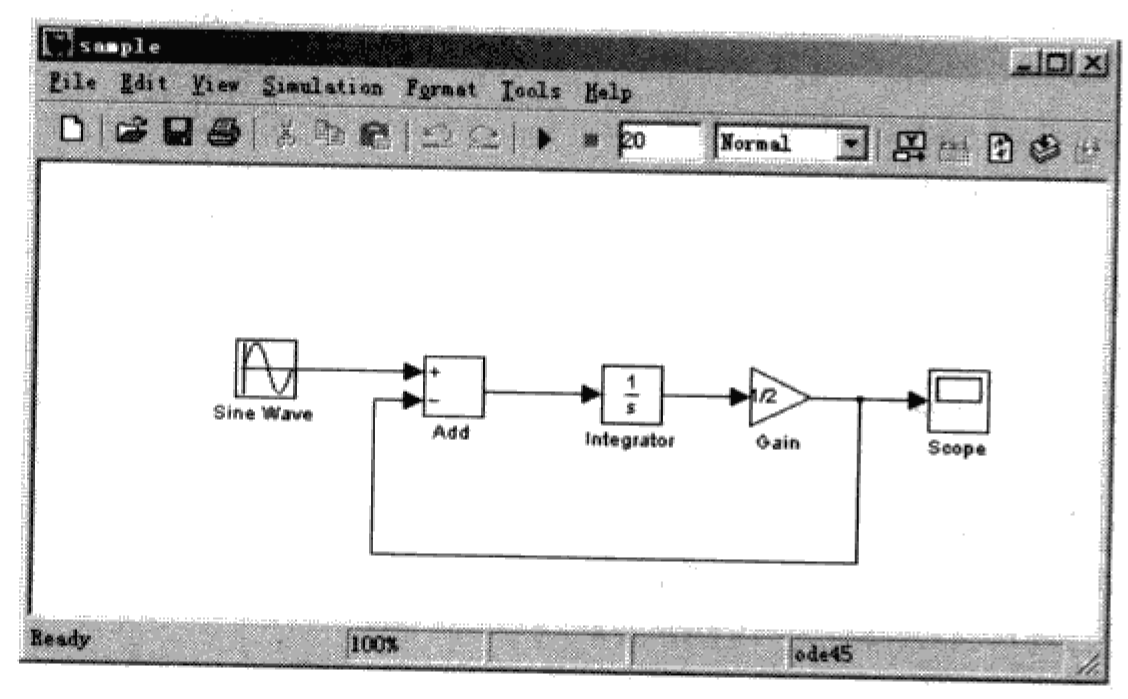


图 8.14　系统模块

说明　Simulink 中各个模块的使用方法，这里就不详细介绍了。

（2）将上面的文件保存为 smple.mdl，并保存在系统的文件夹内。

（3）在命令窗口中输入下面的命令：

```
>>for i=1:4
op(i)=2*i;
opts(i)=simset('initialstate',op(i));
[t,x,y]=sim('sample',[0 10],opts(i));
plot(t,x,'linewidth',2),hold on
grid on
end
```

上面代码运行的结果如图 8.15 所示。

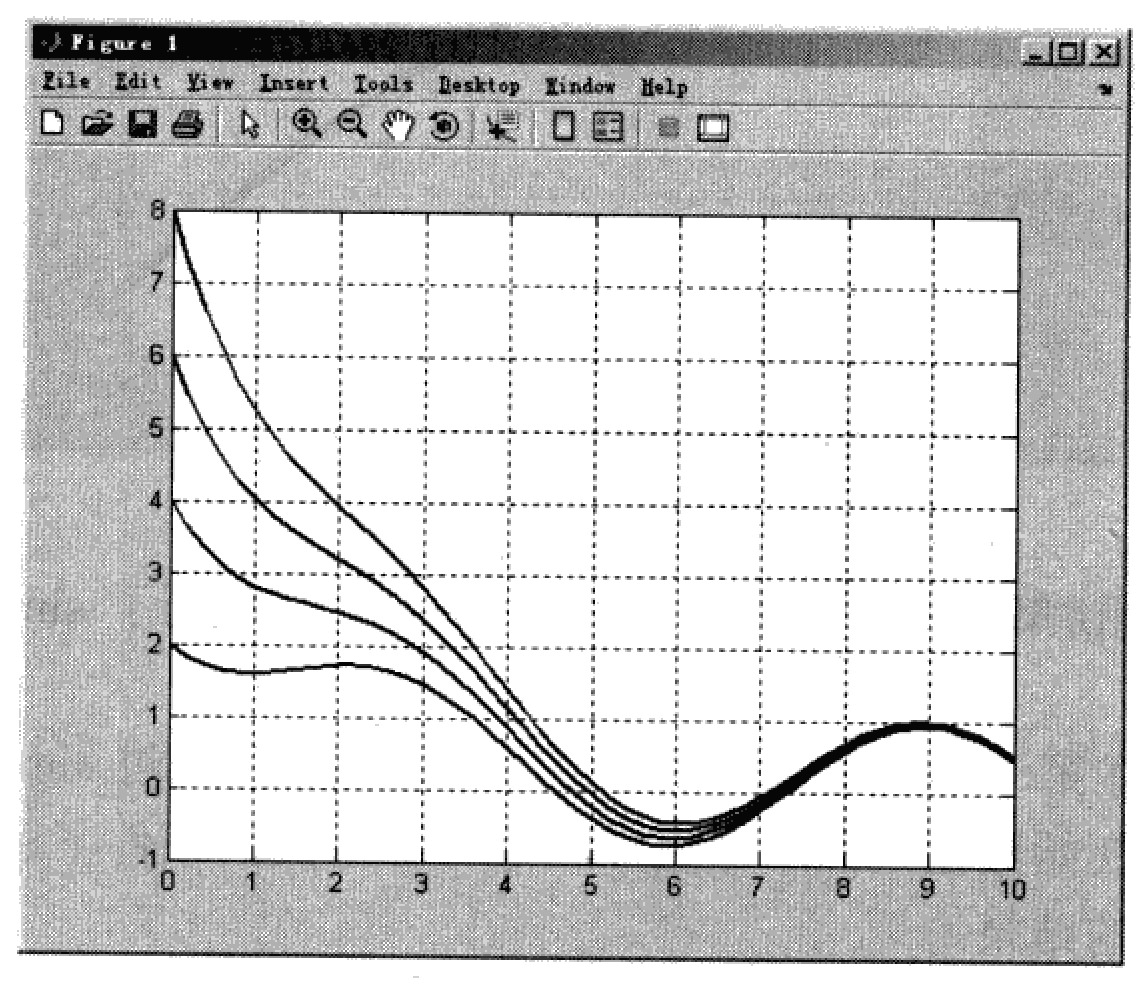


图 8.15　程序代码运行的结果

（4）在命令窗口中输入下面的命令：

```
>> clf;
>> opt(1)=simset('solver','ode15s');
>> [t,x,y]=sim('sample',[0 10],opt(1));
>> plot(t,x,'r','Linewidth',2);hold on;
>> opt(2)=simset('solver','ode45');
>> [t,x,y]=sim('sample',[0 10],opt(2));
```

```
>> plot(t,x,'b','Linewidth',2);
>> axis([9 9.5 0.75 0.95])
>> grid on
```

上面代码运行的结果如图 8.16 所示。

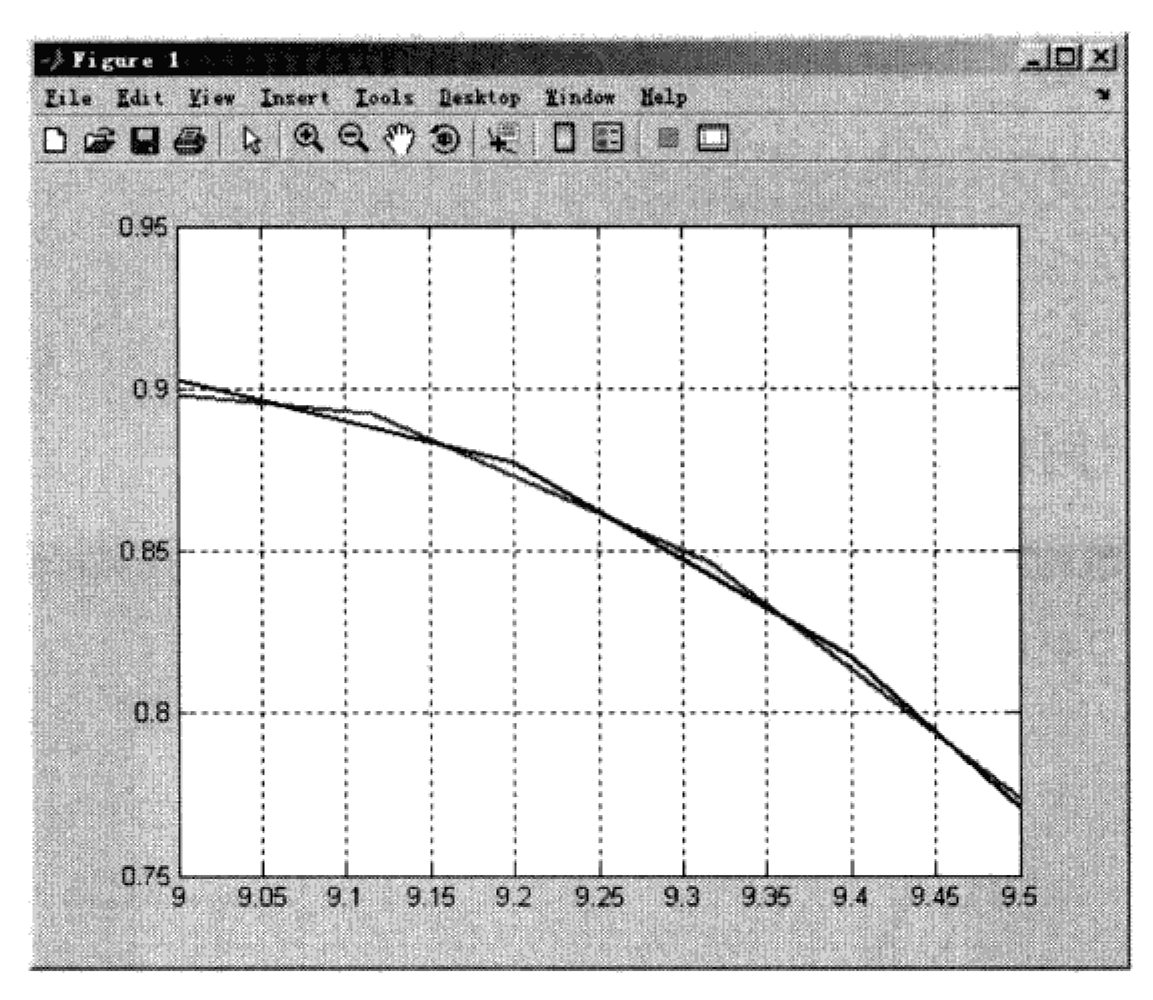


图 8.16　程序结果

【实例讲解】 在上面的步骤中，用户综合使用了 sim 和 simset 命令。

- 在第一个图形中，用户使用循环结构，通过 simset 修改不同的积分初始数值，然后通过 sim 命令在不同的条件下运行系统，得到不同的结果。
- 在第二个图形中，通过 simset 修改解法器，求解同一个仿真系统。然后通过程序命令对比两个不同解法器运行的结果。

8.2.4　linmod 命令——模型的线性化

【语法说明】

- [a,b,c,d]=linmod('sys')：将仿真模型线性化。
- [a,b,c,d]=linmod('sys',x,u)：指定状态、输入向量，将仿真模

型线性化。

- [a,b,c,d]=linmod('sys',x,u,para)：通过设定非线性系统的参数，将仿真模型线性化。

【功能介绍】 模型的线性化。

【实例 8.15】 将非线性模型线性化，并分析线性化的结果。

（1）创建一个简单的非线性模型，其模型框架图如图 8.17 所示。

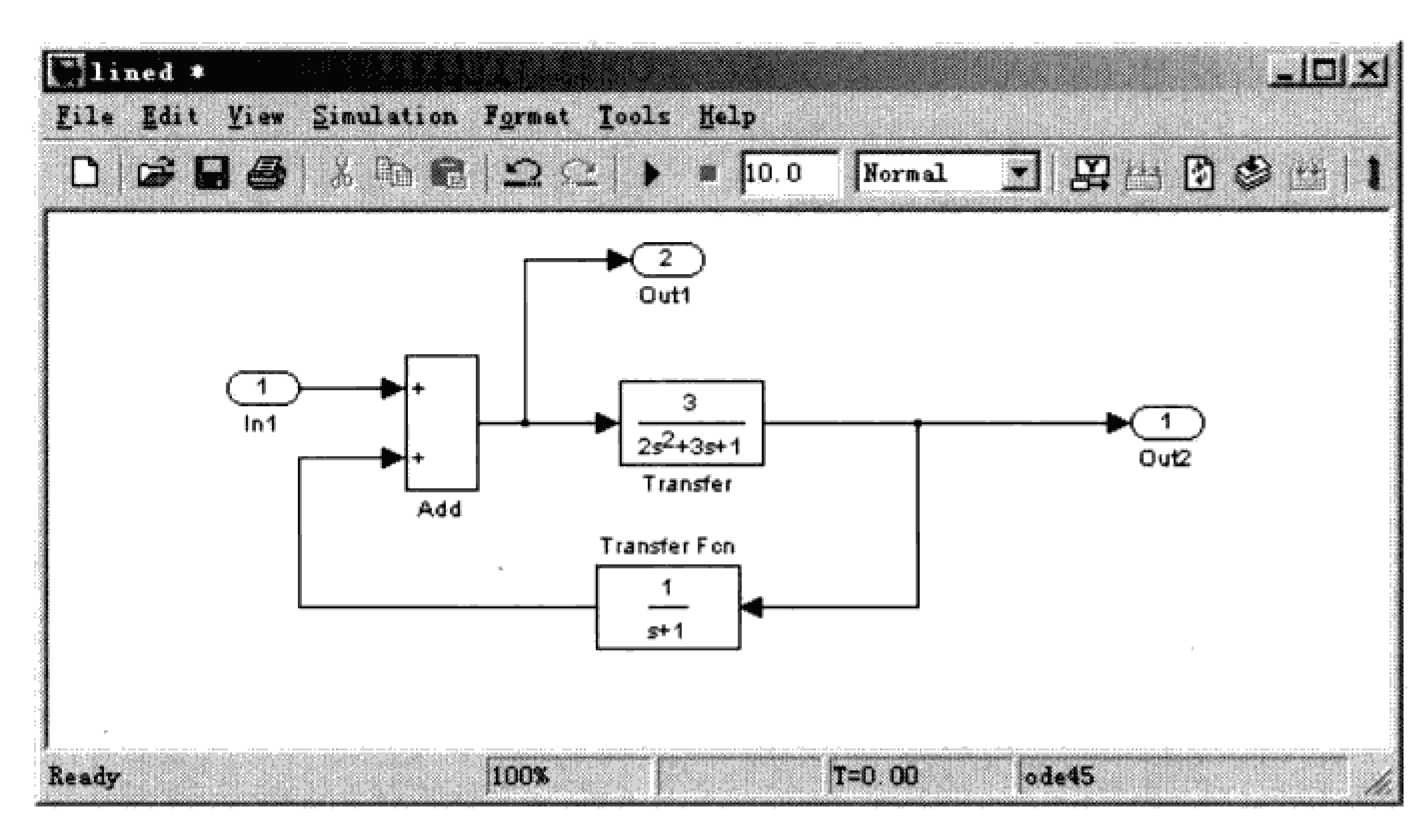


图 8.17 非线性模型

说明 关于上面线性模型的特点，这里就不详细分析了。

（2）进行线性化操作，并得到线性模型的参数。

```
>> [a,b,c,d]=linmod('lined')
```

得到的结果如下：

```
a =
   -1.5000   -0.5000    1.0000
    1.0000         0         0
         0    1.5000   -1.0000
b =
     1
     0
     0
```

```
c =
         0    1.5000         0
         0         0    1.0000
d =
     0
     1
```

（3）将系统的状态空间转换为 LTI 对象。在命令窗口输入下面的命令：

```
>> sys=ss(a,b,c,d)
```

得到的结果如下：

```
a =
          x1     x2    x3
   x1   -1.5   -0.5     1
   x2      1      0     0
   x3      0    1.5    -1
b =
        u1
   x1    1
   x2    0
   x3    0
c =
        x1    x2   x3
   y1    0   1.5    0
   y2    0     0    1
d =
        u1
   y1    0
   y2    1
Continuous-time model.
```

说明　使用上面的命令，用户需要安装 Control System Toolbox。

（4）绘制系统的波特相位振幅图。在命令窗口输入下面的命令：

```
>> bode(sys)
```

上面的命令得到的结果如图 8.18 所示。

（5）绘制单位阶跃和脉冲响应图表。在命令窗口输入下面的命令：

```
>> subplot(1,2,1);step(sys);grid on
>> subplot(1,2,2);impulse(sys);grid on
```

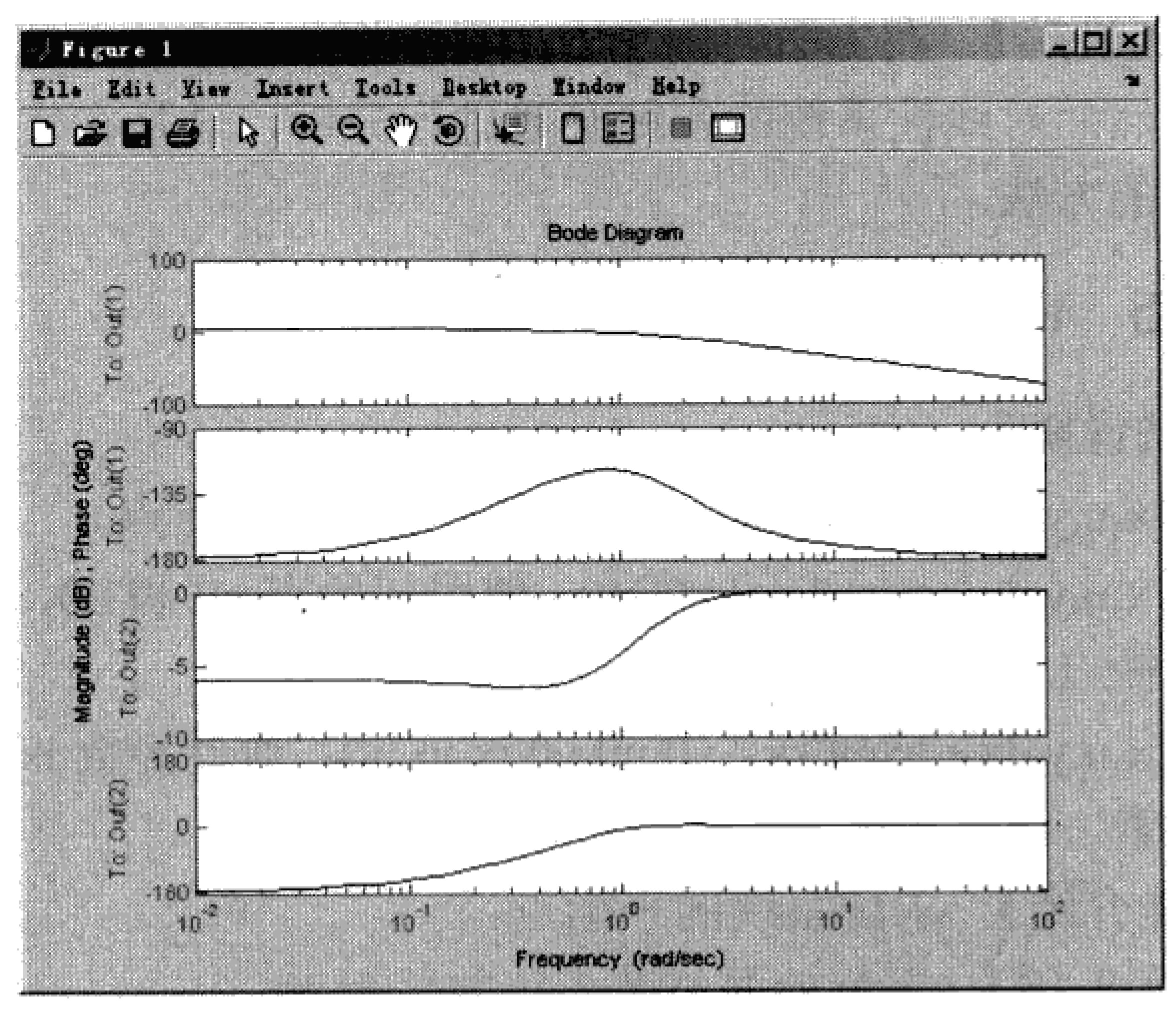


图 8.18 相位振幅图

上面的程序代码得到的结果如图 8.19 所示。

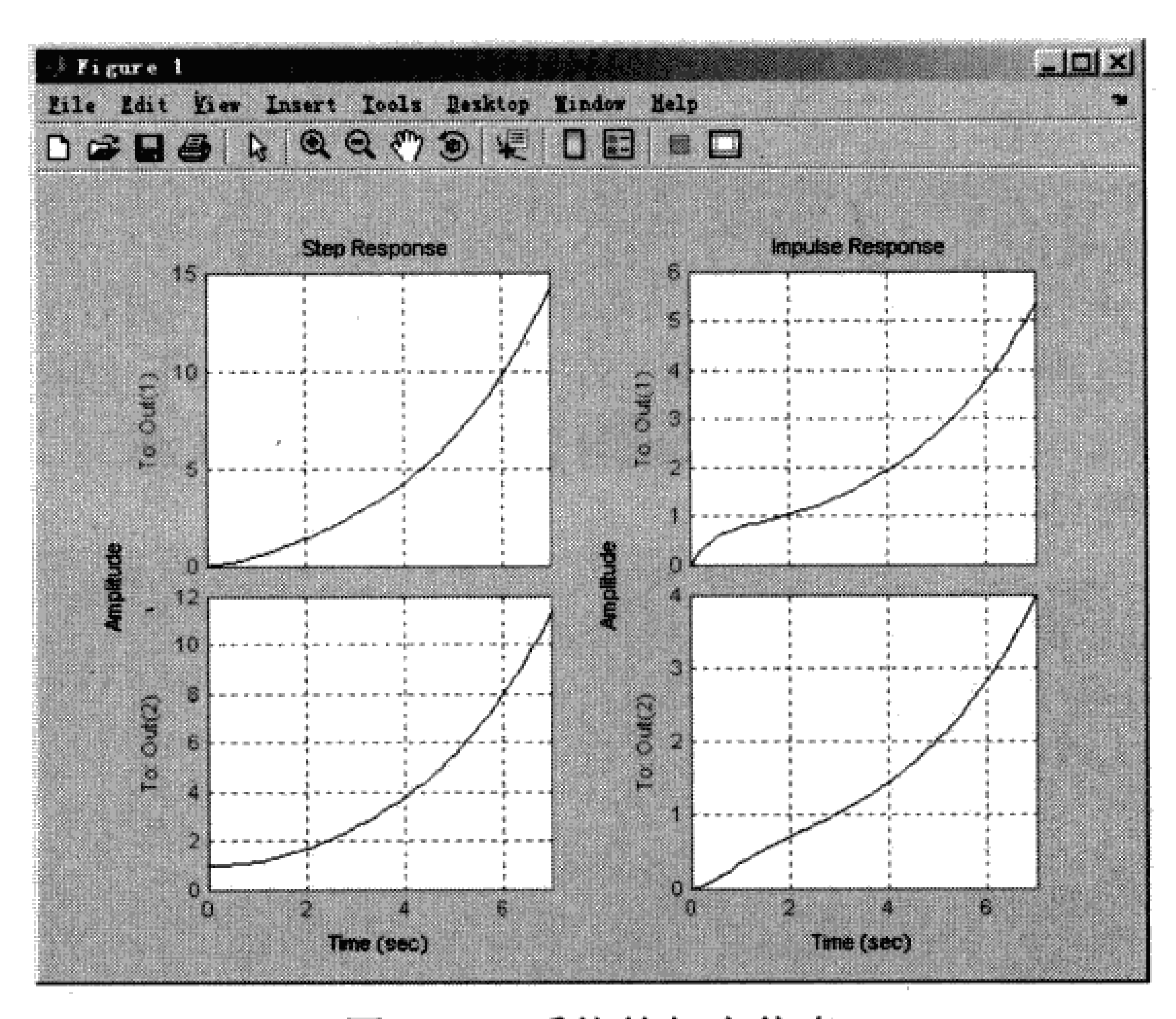


图 8.19 系统的相应信息

【实例讲解】 在上面的例子中，用户将非线性的系统转换为线性系统，然后对线性系统的各种参数进行分析，这是在实际应用中经常使用的方法。

8.2.5 trim 命令——求解系统的平衡点

【语法说明】

- [x,u,y,dx]=trim('sys')：求解仿真系统的平衡点。
- [x,u,y,dx]=trim('sys',x0,u0)：使用初始点 x0 和 u0，求解仿真系统的平衡点。
- [x,u,y,dx]=trim('sys',x0,u0,y0,ix,iu,iy)：通过参数 ix、iu 和 iy 来指定搜索方向，求解仿真系统的平衡点。

【功能介绍】 求解系统的平衡点。

【实例 8.16】 求解前面小节系统的平衡点。

（1）输入初始状态数据。

```
>> x=[0;0;0];
>> u=0;
>> y=[1;1];
>> ix=[];
>> iy=[];
>> iu=[];
>> iy=[1;2];
```

（2）检测搜索点的平衡性。在命令窗口中输入下面的命令：

```
>>[x,u,y,dx]=trim('lined',x,u,y,ix,iu,iy)
```

得到的结果如下：

```
x =
   -0.0000
    1.0000
    1.5000
u =
   -1.0000
y =
    1.5000
    0.5000
```

```
dx =
  1.0e-014 *
   -0.0056
   -0.0322
    0.1110
```

（3）修改初始状态数据：

```
>> x=[0;1;0];
u=0;
y=[3;4];
ix=[];
iy=[];
iu=[];
iy=[1;2];
```

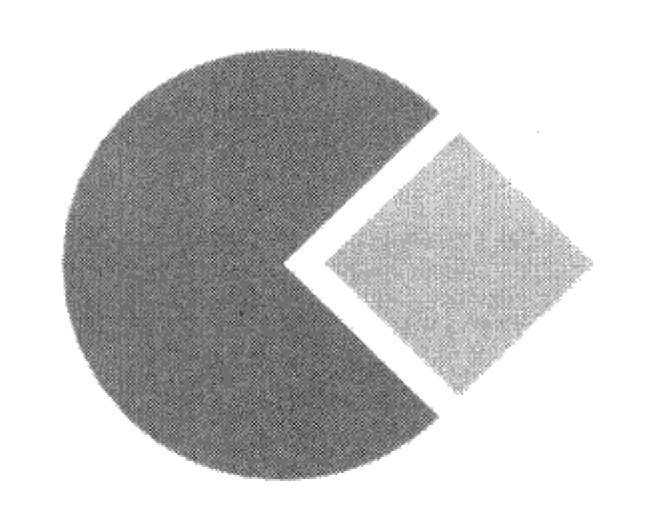
（4）重新检测搜索点的平衡性。在命令窗口中输入下面的命令：

```
>> [x,u,y,dx]=trim('lined',x,u,y,ix,iu,iy)
```

得到的结果如下：

```
x =
   -0.0000
    3.5000
    5.2500
u =
   -3.5000
y =
    5.2500
    1.7500
dx =
  1.0e-015 *
   -0.8882
   -0.0000
    0.8882
```

【实例讲解】 在 MATLAB 中，并不是所有的系统都有平衡点。如果不能搜索到平衡点，trim 命令将会直接返回错误信息。



第9章 图形用户界面制作

用户界面（或接口）是指：人与机器（或程序）之间交互作用的工具和方法。如键盘、鼠标、跟踪球、话筒都可成为与计算机交换信息的接口。图形用户界面（Graphical User Interfaces，GUI）则是由窗口、光标、按键、菜单、文字说明等对象（Objects）构成的一个用户界面。用户通过一定的方法（如鼠标或键盘）选择、激活这些图形对象，使计算机产生某种动作或变化。

假如读者所从事的数据分析、解方程、计算结果可视工作比较单一，那么一般不会考虑 GUI 的制作。但是如果读者想向别人提供应用程序，想进行某种技术、方法的演示，想制作一个供反复使用且操作简单的专用工具，那么图形用户界面也许是最好的选择之一。

9.1 入门

本节将通过一个简单的实例来阐述图形用户界面 GUI 的用法和用途。

【实例 9.1】 建立一个图形界面来显示和处理二维图形的颜色、线型及数据点的图标。要求对应的基本功能是：

- 建立一个主坐标系，用来显示要绘制的二维图形；

- 建立一个列表框，允许用户选择不同的着色方法；
- 建立一组按钮，用于处理二维图形线型和数据点的图标；
- 为图形界面加入菜单项，用于决定坐标轴是否显示及坐标轴上是否需要网格。

下面详细介绍创建的过程。

（1）创建坐标轴，并创建空白图形。在命令窗口输入下面的命令：

```
yWin=figure('Name','MATLAB 绘 图 功 能 演 示 (2007/9/23)',
'Menubar','none');
H=axes('Box','on','Units','points','Position',[100,
20,190,200]);
set(gcf,'currentaxes',H);
str='\fontname{宋体}绘制图形';                    %设置标题文
字
text(0.12,0.93,str,'fontsize',13); %设置文本的样式
h_fig=get(H,'parent');
set(h_fig,'unit','normalized','position',[0.1,0.2,
0.7,0.4]);
```

得到的图形如图 9.1 所示。

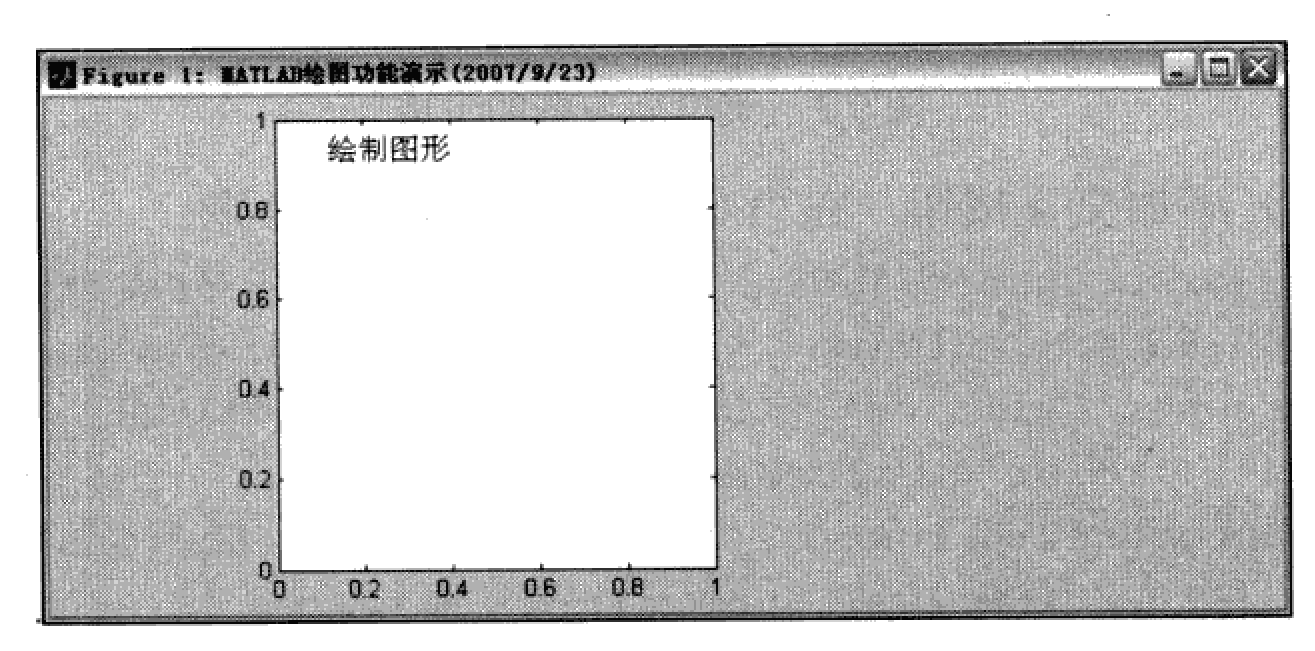


图 9.1　图形界面生成

（2）创建菜单列表。在命令窗口输入下面的命令：

```
ymenu001=uimenu(yWin,'label','Grid');       %制作第一个下
拉列表
ymenu0011=uimenu(ymenu001,'label','Grid
```

```
on','callback','Grid on');    %制作第一个下拉列表的子列表
    ymenu0011=uimenu(ymenu001,'label','Grid
off','callback','Grid off');
    ymenu002=uimenu(yWin,'label','Axes');        %制作第二个
下来列表
    ymenu0021=uimenu(ymenu002,'label','Axes
on','callback','set(gca,''visible'',''on'')');
    ymenu0021=uimenu(ymenu002,'label','Axes
off','callback','set(gca,''visible'',''off'')');
```

得到的图形如图 9.2 所示。

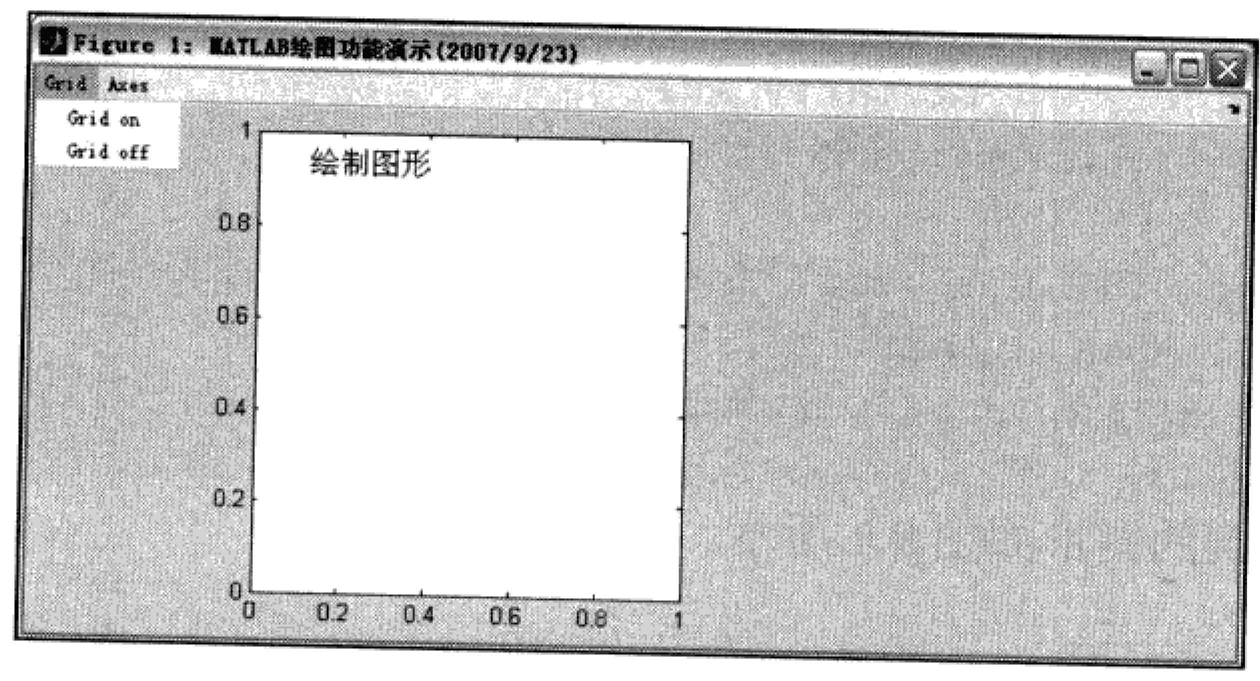


图 9.2　菜单列表框的生成

（3）添加交互按钮。在命令窗口输入下面的命令：

```
    >>CBox=uicontrol(yWin,'style','listbox','Position',
[600,240,80,45],'String','red|blue|black|bronze|mauve|ye
llow|green','Callback',['Colornum=get(CBox,''value'');',
'set(Fun,''color'',ColorStr(Colornum))']);
    pushbutton1=uicontrol(yWin,'style','pushbutton','Pos
ition',[600,20,80,20],'String','实　线',...
    'Callback',['Fun=plot(x,sin(x),union(ColorStr(Colorn
um),LineStr(1)))']);
    pushbutton2=uicontrol(yWin,'style','pushbutton','Pos
ition',[600,50,80,20],'String','虚　线',...
    'Callback',['Fun=plot(x,sin(x),union(ColorStr(Colorn
um),LineStr(2)))']);
```

```
pushbutton3=uicontrol(yWin,'style','pushbutton','Position',[600,80,80,20],'String','星  号',...
'Callback',['Fun=plot(x,sin(x),union(ColorStr(Colornum),LineStr(3)))']);
pushbutton4=uicontrol(yWin,'style','pushbutton','Position',[600,110,80,20],'String','圆  圈',...
'Callback',['Fun=plot(x,sin(x),union(ColorStr(Colornum),LineStr(4)))']);
pushbutton5=uicontrol(yWin,'style','pushbutton','Position',[600,140,80,20],'String','五角星',...
'Callback',['Fun=plot(x,sin(x),union(ColorStr(Colornum),LineStr(5)))']);
pushbutton6=uicontrol(yWin,'style','pushbutton','Position',[600,170,80,20],'String','十字号',...
'Callback',['Fun=plot(x,sin(x),union(ColorStr(Colornum),LineStr(6)))']);
pushbutton7=uicontrol(yWin,'style','pushbutton','Position',[600,200,80,20],'String','六角星',...
'Callback',['Fun=plot(x,sin(x),union(ColorStr(Colornum),LineStr(7)))']);
```

得到的图形如图 9.3 所示。

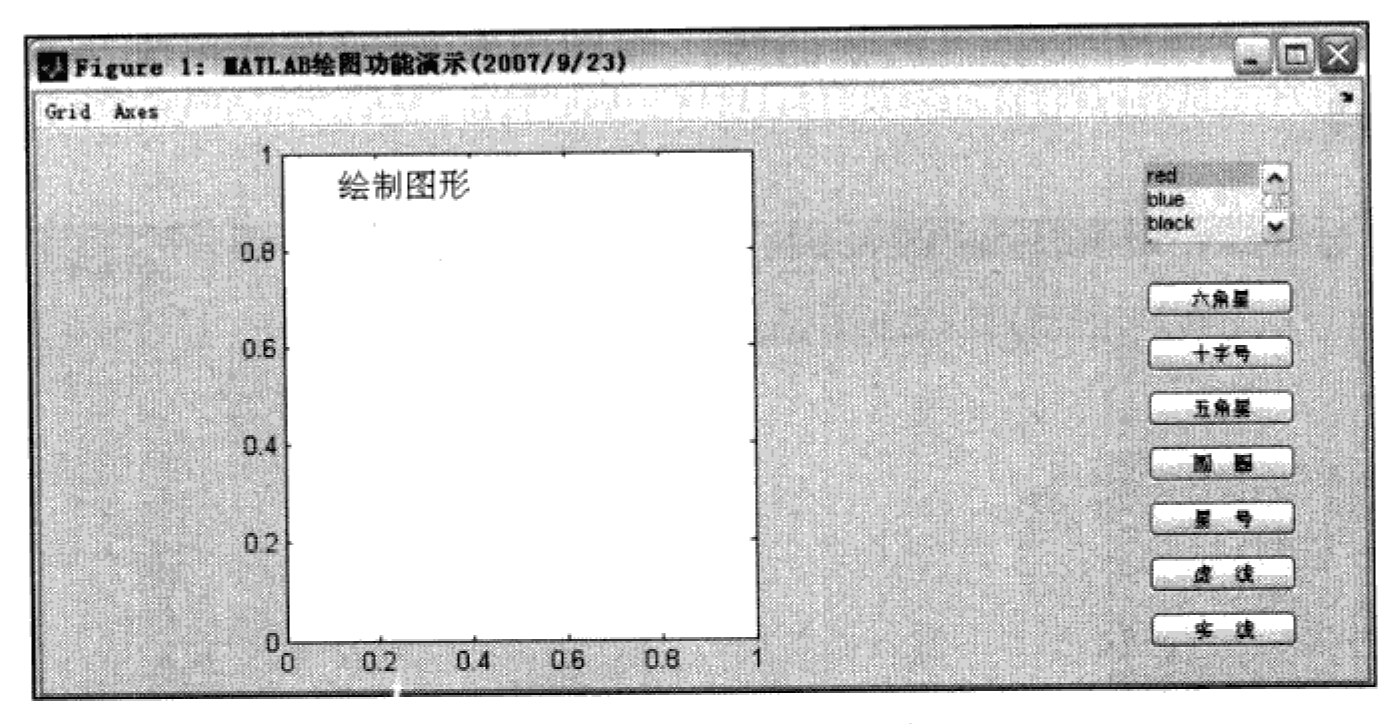


图 9.3 创建按钮控件

（4）绘制基础图表。在命令窗口输入下面的命令：

```
ColorStr=['r','b','k','c','m','y','g'];
```

```
    LineStr=['-',':','*','o','p','+','h'];
    Colornum=1;
    x=0:0.1:2*pi;
    Fun=plot(x,sin(x),union(ColorStr(Colornum),LineStr(1
)));
```

得到的图形如图 9.4 所示。

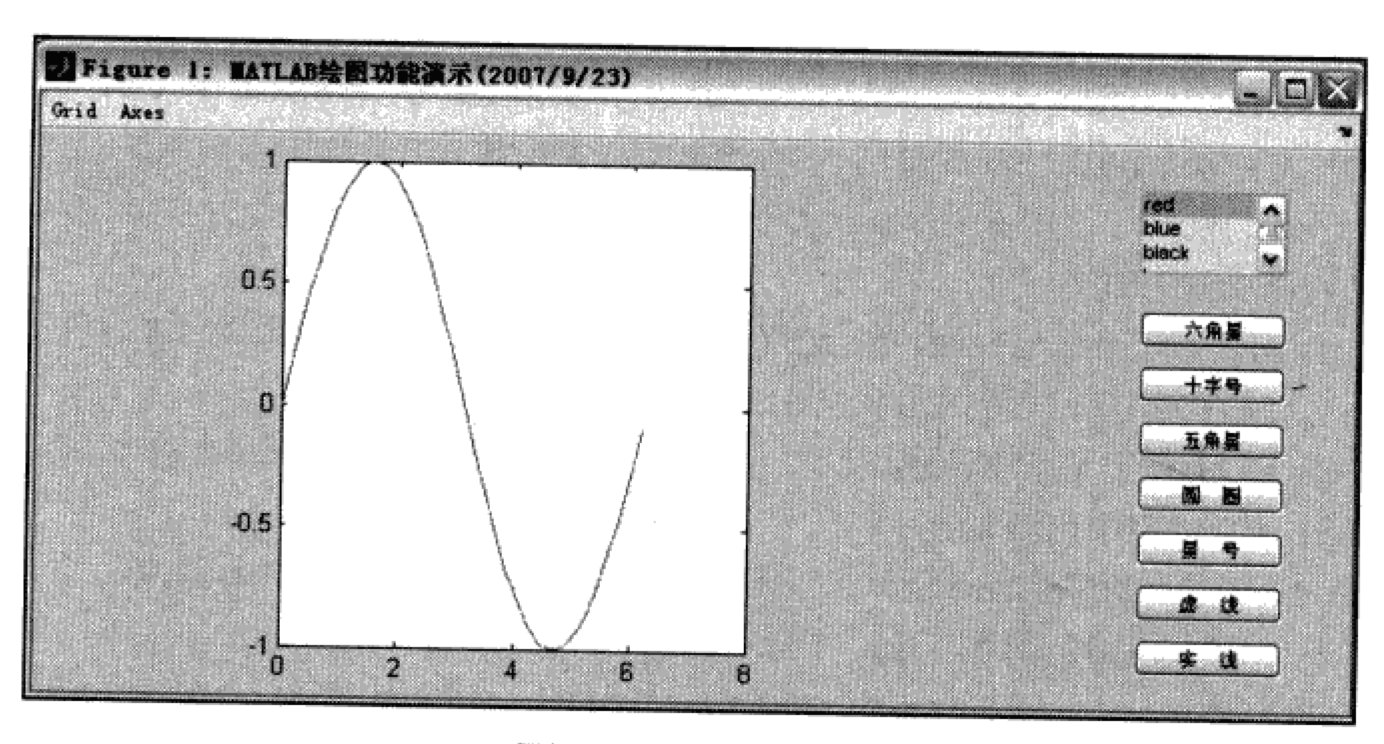


图 9.4 绘制曲线

下面通过调整颜色属性和右边的按钮控制曲线的显示。如图 9.5 所示为蓝色圆圈线型的正弦曲线，而图 9.6 所示为紫色十字型的正弦曲线。

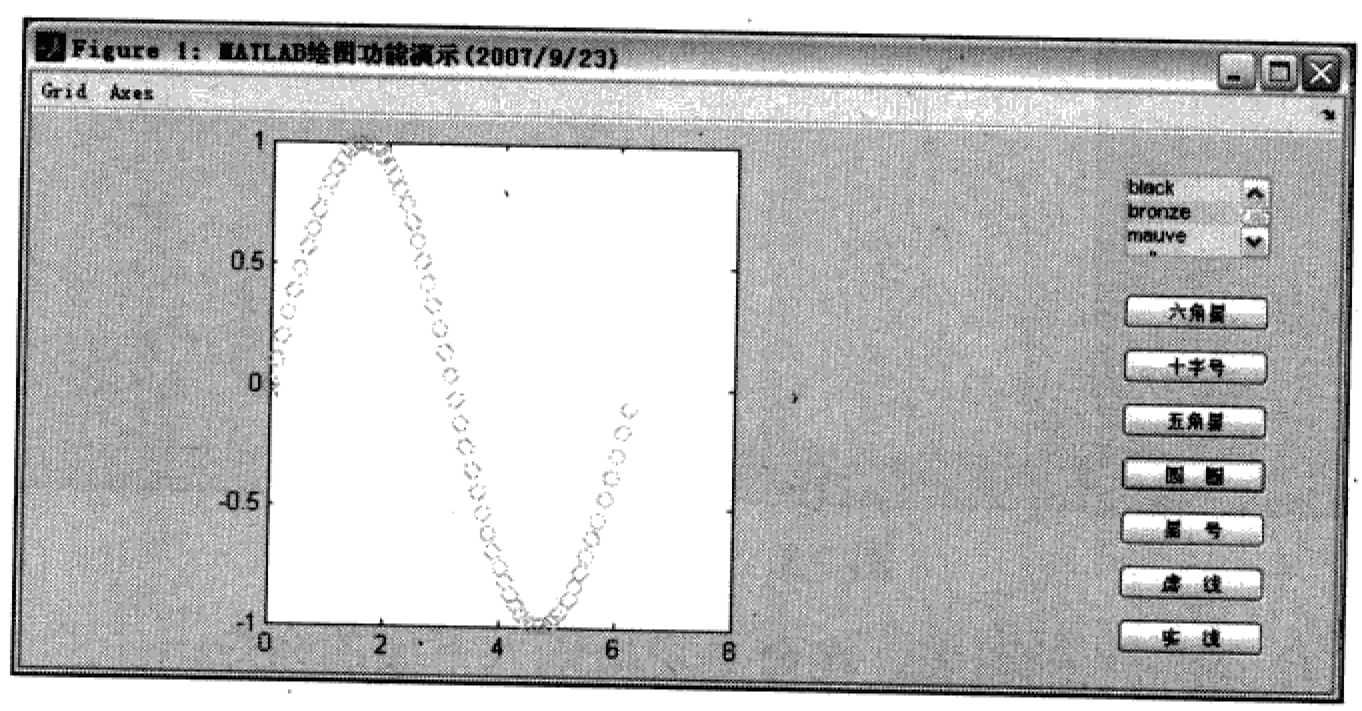


图 9.5 蓝色圆圈线型的正弦曲线

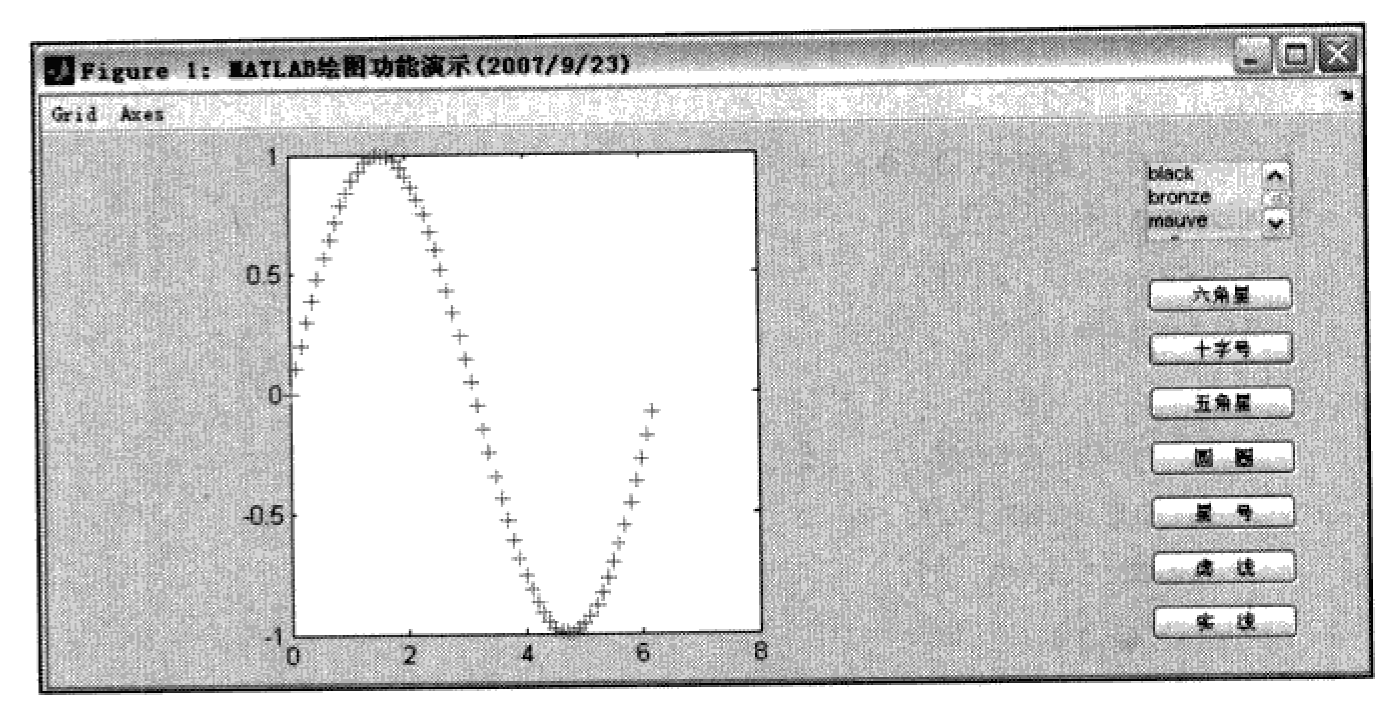


图 9.6　紫色十字型的正弦函数

9.2　图形用户界面设计的基本函数

9.2.1　get 函数——获得对象属性

【语法说明】

- get(H)：返回句柄 H 的属性值和当前属性值。
- V=get(H)：返回句柄 H 的属性值和当前属性值并把相应值放到 V 中。
- get(H，'PropertyName')：返回句柄 H 的某个属性值。

【功能介绍】 获取对象的属性值。

【实例 9.2】 创建一个线条对象，列出其属性和当前属性值。

```
>> H=line(10,30)    %创建一个线条句柄
>> get(H)
```

得到的结果为：

```
Color = [0 0 1]
    EraseMode = normal
    LineStyle = -
    LineWidth = [0.5]
    Marker = none
```

```
        MarkerSize = [6]
        MarkerEdgeColor = auto
        MarkerFaceColor = none
        XData = [10]
        YData = [30]
        ZData = []

        BeingDeleted = off
        ButtonDownFcn =
        Children = []
        Clipping = on
        CreateFcn =
        DeleteFcn =
        BusyAction = queue
        HandleVisibility = on
        HitTest = on
        Interruptible = on
        Parent = [91.001]
        Selected = off
        SelectionHighlight = on
        Tag =
        Type = line
        UIContextMenu = []
        UserData = []
        Visible = on
>> get(H,'visible') %观测对象的可视状态
```

得到的结果为：

```
ans =
on
```

【实例讲解】 通过 get 函数可以得到这个对象的所有属性。get(H,'visible')只是提取一个属性。

9.2.2 set 函数——设置对象属性

【语法说明】 set(handle，'Property Name'，Property Value)。

【功能介绍】 修改和设定句柄图形对象。

【实例 9.3】 创建一个窗口，列出可设置的属性和当前属性。

```
>> H=figure
H=
    1
>> set(H)
```

得到的结果为：

```
Alphamap
    BackingStore: [ {on} | off ]
    CloseRequestFcn: string -or- function handle -or-
cell array
    Color
    Colormap
    CurrentAxes
    CurrentCharacter
    CurrentObject
    CurrentPoint
    DockControls: [ {on} | off ]
    DoubleBuffer: [ {on} | off ]
    FileName
    IntegerHandle: [ {on} | off ]
    InvertHardcopy: [ {on} | off ]
    KeyPressFcn: string -or- function handle -or- cell
array
    MenuBar: [ none | {figure} ]
    MinColormap
    Name
    NextPlot: [ new | {add} | replace | replacechildren ]
    NumberTitle: [ {on} | off ]
    PaperUnits: [ {inches} | centimeters| normalized |
points ]
    PaperOrientation: [ {portrait} | landscape |
rotated ]
    PaperPosition
    PaperPositionMode: [ auto | {manual} ]
    PaperSize
    PaperType: [ {usletter} | uslegal | A0 | A1 | A2 |
A3 | A4 | A5 | B0 | B1 | B2 | B3 | B4 | B5 | arch-A | arch-B|
arch-C | arch-D | arch-E | A | B | C | D | E | tabloid | <custom> ]
    Pointer: [ crosshair | fullcrosshair | {arrow} |
ibeam | watch | topl | topr | botl | botr | left | top | right
```

```
| bottom | circle | cross | fleur | custom ]
        PointerShapeCData
        PointerShapeHotSpot
        Position
        Renderer: [ {painters} | zbuffer | OpenGL | None ]
        RendererMode: [ {auto} | manual ]
        Resize: [ {on} | off ]
        ResizeFcn: string -or- function handle -or- cell
array
        SelectionType: [ normal | open | alt | extend ]
        ShareColors: [ {on} | off ]
        ToolBar: [ none | {auto} | figure ]
        Units: [ inches | centimeters | normalized | points
| {pixels} | characters ]
        WindowButtonDownFcn: string -or- function handle
-or- cell array
        WindowButtonMotionFcn: string -or- function handle
-or- cell array
        WindowButtonUpFcn: string -or- function handle -or-
cell array
        WindowStyle: [ {normal} | modal | docked ]
        WVisual: { 00 (RGB 16 GDI, Bitmap, Window) }
     01 (RGB 16 bits(05 06 05 00) zdepth  0, Hardware
Accelerated, Opengl, Window)
     02 (RGB 16 bits(05 06 05 00) zdepth  0, Hardware
Accelerated, Opengl, Double Buffered, Window)
     03 (RGB 16 bits(05 06 05 00) zdepth 16, Hardware
Accelerated, Opengl, Window)
     04 (RGB 16 bits(05 06 05 00) zdepth 16, Hardware
Accelerated, Opengl, Double Buffered, Window)
     05 (RGB 16 bits(05 06 05 00) zdepth 24, Hardware
Accelerated, Opengl, Window)
     06 (RGB 16 bits(05 06 05 00) zdepth 24, Hardware
Accelerated, Opengl, Double Buffered, Window)
     07 (RGB 16 bits(05 06 05 00) zdepth 32, Generic, Opengl,
GDI, Bitmap, Window)
     08 (RGB 16 bits(05 06 05 00) zdepth 16, Generic, Opengl,
GDI, Bitmap, Window)
     09 (RGB 16 bits(05 06 05 00) zdepth 32, Generic, Opengl,
Double Buffered, Window)
```

```
      10 (RGB 16 bits(05 06 05 00) zdepth 16, Generic, Opengl,
Double Buffered, Window)
      11 (RGB 16 bits(05 06 05 08) zdepth 32, Generic, Opengl,
GDI, Bitmap, Window)
      12 (RGB 16 bits(05 06 05 08) zdepth 16, Generic, Opengl,
GDI, Bitmap, Window)
      13 (RGB 16 bits(05 06 05 08) zdepth 32, Generic, Opengl,
Double Buffered, Window)
      14 (RGB 16 bits(05 06 05 08) zdepth 16, Generic, Opengl,
Double Buffered, Window)

        WVisualMode: [ {auto} | manual ]

        ButtonDownFcn: string -or- function handle -or-
cell array
        Children
        Clipping: [ {on} | off ]
        CreateFcn: string -or- function handle -or- cell
array
        DeleteFcn: string -or- function handle -or- cell
array
        BusyAction: [ {queue} | cancel ]
        HandleVisibility: [ {on} | callback | off ]
        HitTest: [ {on} | off ]
        Interruptible: [ {on} | off ]
        Parent
        Selected: [ on | off ]
        SelectionHighlight: [ {on} | off ]
        Tag
        UIContextMenu
        UserData
        Visible: [ {on} | off ]
```

【实例讲解】 上面列出了对象的所有属性，用户可以根据自己的需要选择其中的一个或几个属性设置。

【实例 9.4】 将图形对象的颜色设为蓝色。

```
H =figure
set(H,'color','b')
```

得到的图形如图 9.7 所示。

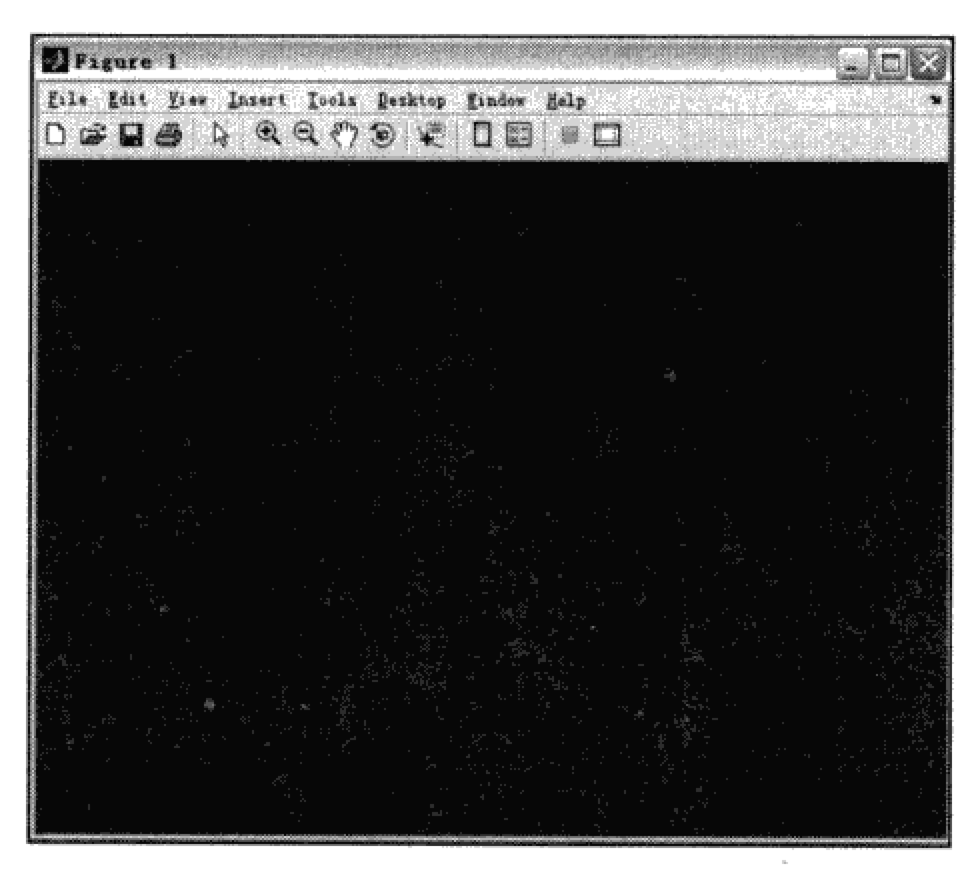


图 9.7　图形对象颜色设置

【实例讲解】 上面这个例子演示了基本的颜色设置属性。

9.2.3　gcf 函数——回归当前图形句柄

【语法说明】 gcf()：返回当前图形的句柄值。

【功能介绍】 获得当前图形的句柄值。

【实例 9.5】 返回图形的句柄值。

```
>> H=figure
H =

    1

>> gcf()

ans =

    1
```

【实例讲解】 获取句柄的函数，通常会结合图形函数一起使用。

9.2.4　figure 函数——图形窗口的建立

【语法说明】 H=figure（属性 1，属性值 1，属性 2，属性值 2，…）。

【功能介绍】 建立新的窗口。

【实例 9.6】 建立一个图形，先隐藏起来不让其显示，然后设置属性并显示。

```
>>  H=figure('Visible','off')

H =

     1

>>   set(H,'color',[1,1,1],'position',[0,0,400,300],
'name','example')
>> set(H,'Visible','on');
```

得到的结果如图 9.8 所示。

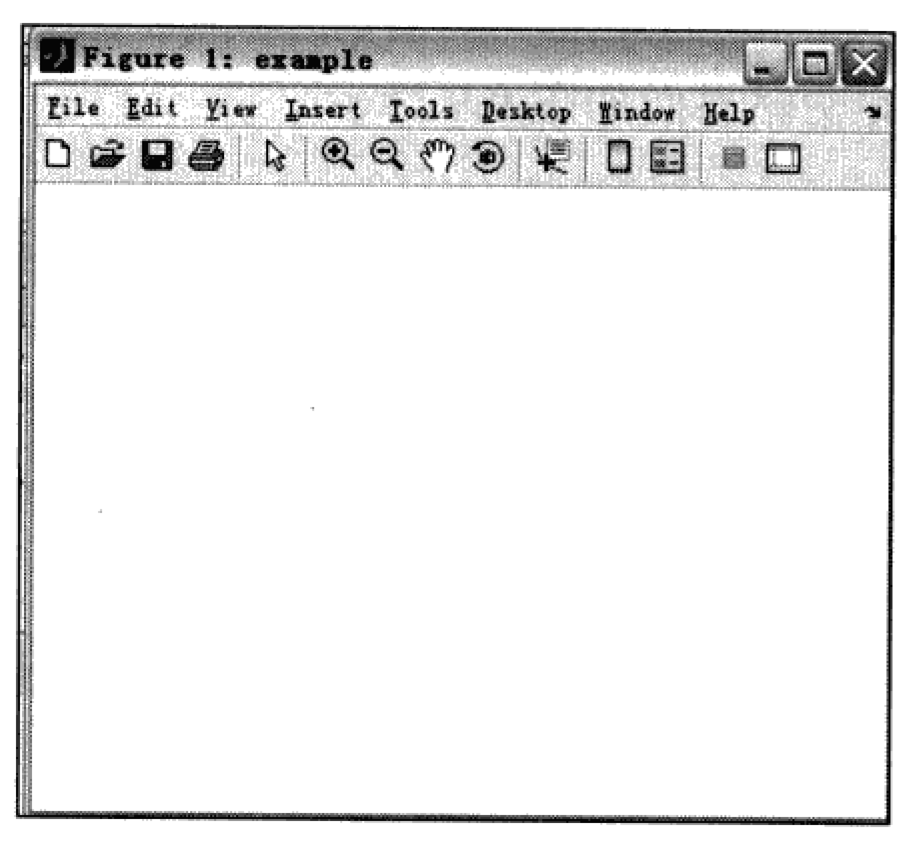


图 9.8　建立新的图形

【实例讲解】 上例中图形的名称就为“example”；背景颜色为[1,1,1]的白色；窗口位置属性‘position’为[0,0,400,300]，其起点为屏幕左下角，宽度为 300 个像素，高度为 400 个像素。

9.2.5　uimenu 函数——自制用户菜单的创建

【语法说明】 Menu_name=uimenu(gcf,'label','name')。

【功能介绍】 自制用户菜单。

【实例 9.7】 自制一个带下拉菜单的用户菜单。该菜单能使图形窗背景颜色设置为兰色、红色和绿色。

```
figure
h_menu=uimenu(gcf,'label','Color');                    %制作主菜单
h_submenu1=uimenu(h_menu,'label','Blue',...
    'callback','set(gcf,''Color'',''blue'')');   %建立子菜单 1
h_submenu2=uimenu(h_menu,'label','Red',...         %建立子菜单 2
    'callback','set(gcf,''Color'',''red'')');
h_submenu3=uimenu(h_menu,'label','Green',...   %建立子菜单 3
    'callback','set(gcf,''Color'',''green'')');
```

上例产生的初始界面如图 9.9 所示。

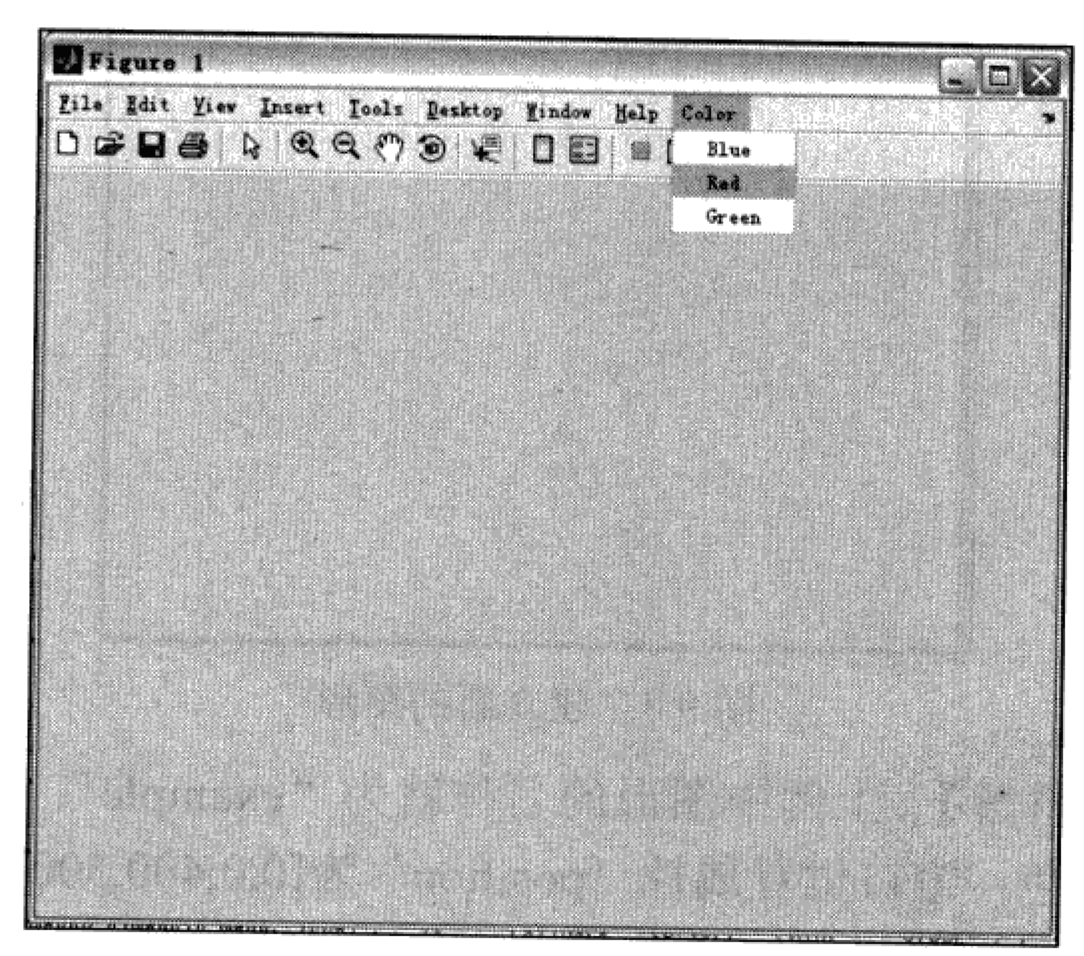


图 9.9 Color 菜单项

当单击【Red】按钮时，产生的界面如图 9.10 所示。

当单击【Blue】按钮时，产生的界面如图 9.11 所示。

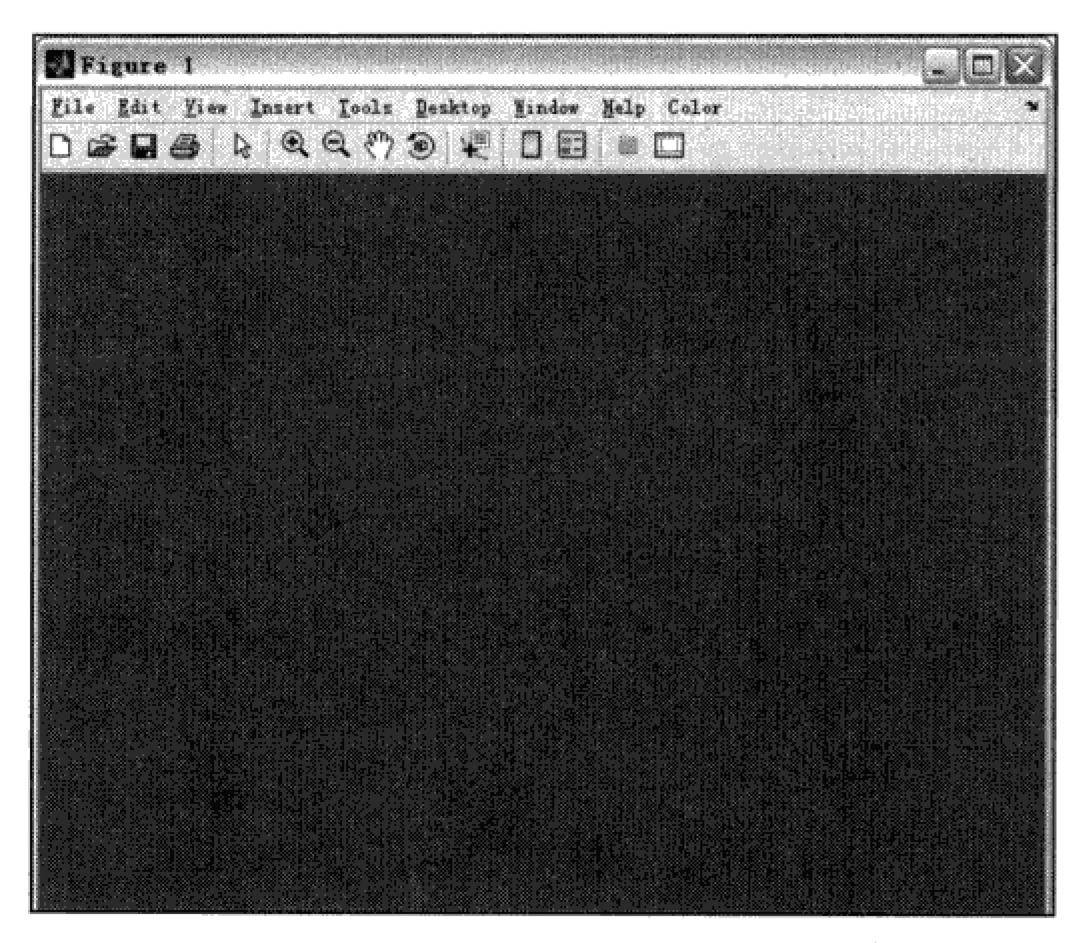


图 9.10　红色界面显示

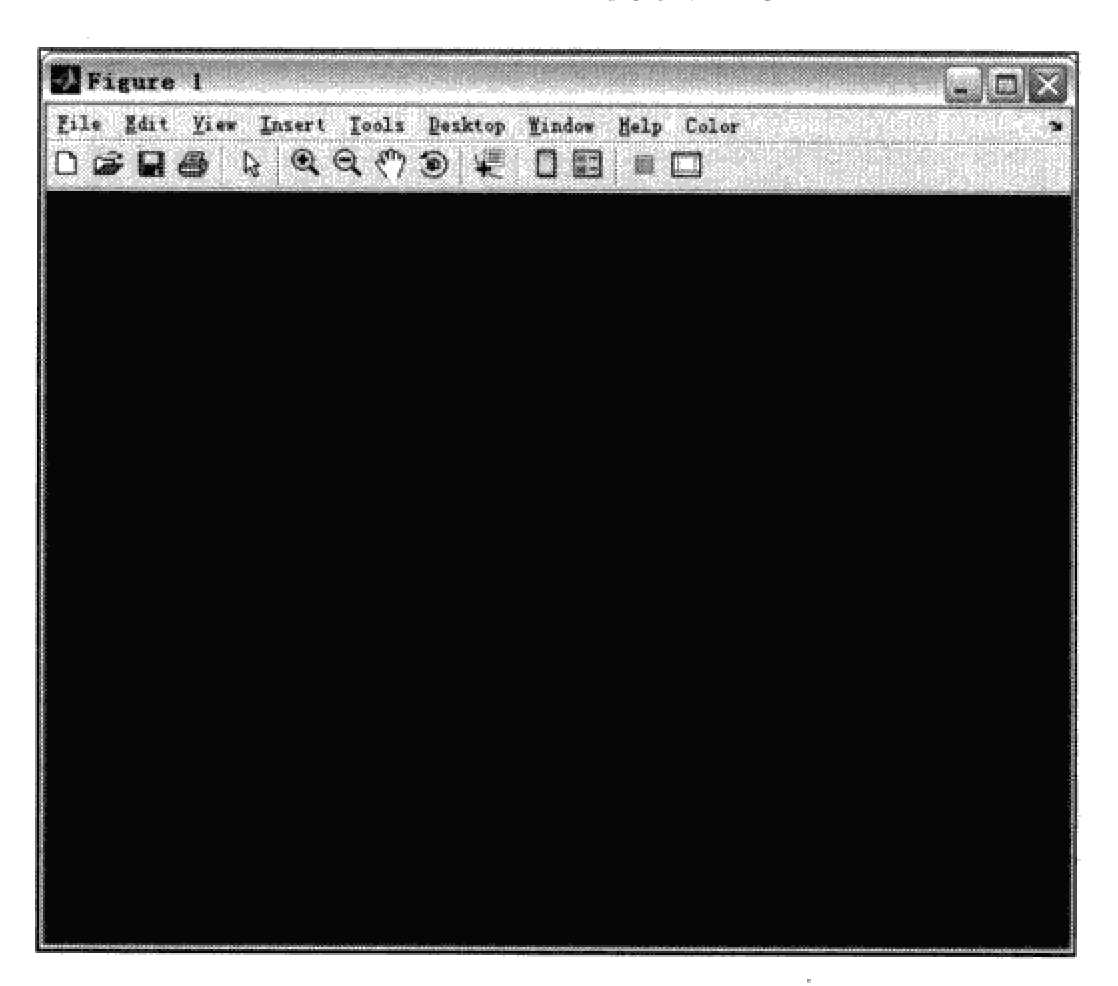


图 9.11　蓝色界面显示

【实例讲解】 有其他编程经验的用户可以发现，在 MATLAB 中创建菜单十分便利。

【实例 9.8】 建立一个带下拉菜单和不带下拉菜单的窗口。

```
>> H=figure('Color',[1 1 1],'position',[0,0,400, 300],
'Name','example','NumberTitle','off','MenuBar','none')

H =

     1

>> mfile=uimenu(H,'label','&File');
>> mtransform=uimenu(H,'label','&Transforms');
>> mhelp=uimenu(H,'label','Help');
>>     uimenu(mfile,'label','&New','call','disp(''new
file selected'')');
>>   uimenu(mfile,'label','&Open','call','disp(''open
file selected'')');
>> msave=uimenu(mfile,'label','Save','Enable','off');
```

产生的界面如图 9.12 所示。

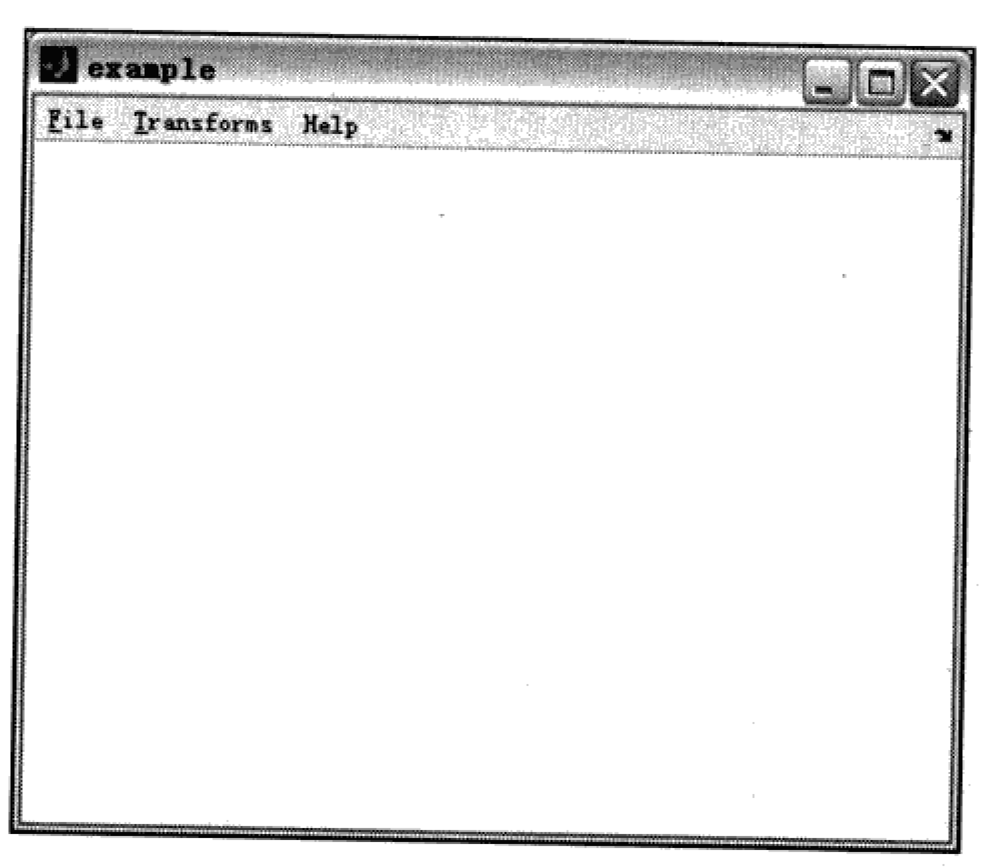


图 9.12　创建子菜单

【实例讲解】 这个例子首先产生一个定制的用户图形窗口，其窗口的颜色为白色，窗口的初始位置为起点于屏幕的左下角，宽度为 400 个像素，高度为 300 个像素。标题为“example”，取消图形的编号，取消默认菜单。然后建立菜单项“File”，“Transform”，

“Help”，最后又建立了一个“Save”菜单使其隐藏。而后为“File”菜单新建两个子菜单。

9.2.6 设置快捷键

【实例 9.9】 使上面设计的 Color 菜单项及其下拉的 Blue 菜单各带一个快捷键，而另一项下拉菜单 Red 带一个快捷键。

```
>>  H=figure('Color',[1 1 1],'position',[0,0,400, 300],
'Name','example','NumberTitle','off','MenuBar','none')

H =

     1

>> mfile=uimenu(H,'label','&File');
>>  mtransform=uimenu(H,'label','&Transforms');
>> mhelp=uimenu(H,'label','Help');
>>     uimenu(mfile,'label','&New','call','disp(''new
file selected'')','Accelerator','n');
>>    uimenu(mfile,'label','&New','call','disp(''Open
file selected'')','Accelerator','o');
```

产生的界面如图 9.13 所示。

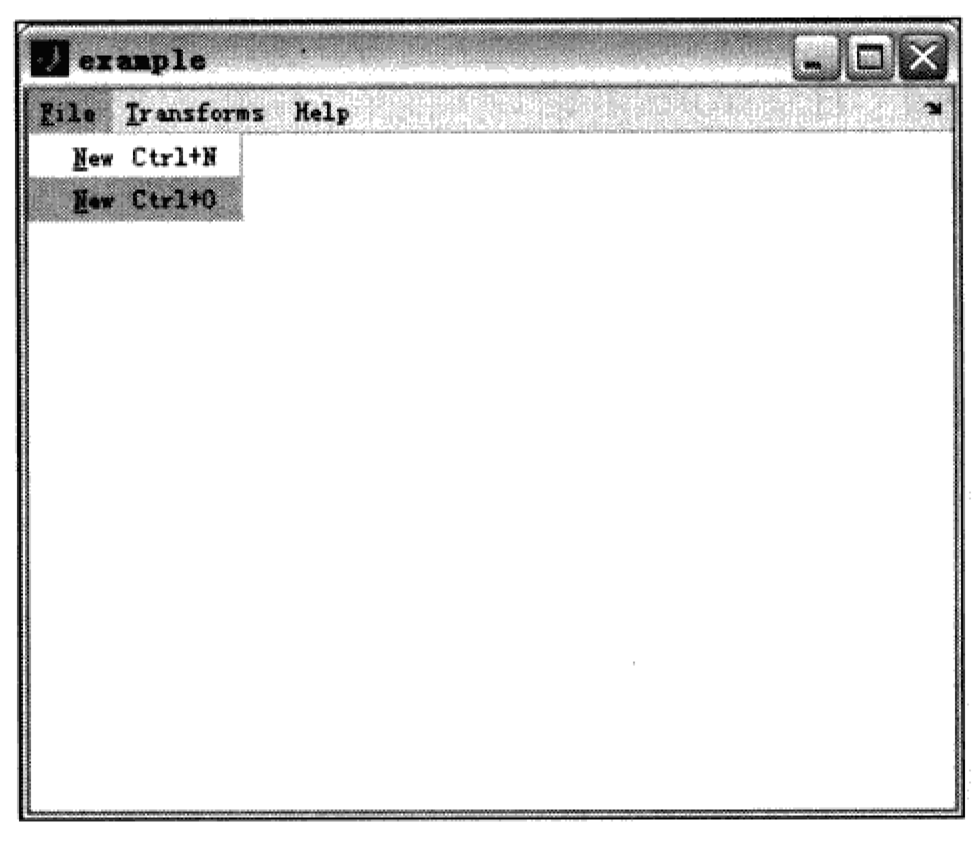


图 9.13 快捷键设计

【实例讲解】 如果要在哪个菜单项设置快捷键，只需要在原来的菜单属性中添加“Accelerator”属性，并在其后填写相应的快捷键字母即可。

9.2.7 helpdlg 函数——帮助窗口对话框

【语法说明】 H=helpdlg(帮助信息,标题)。

【功能介绍】 用来创建一个帮助窗口，在这个有标题的窗口中显示帮助信息。如果具有相同标题的窗口已经存在，则将其调至最上层；如果不存在，则新建一个。用户可以单击“OK”按钮取消帮助窗口继续以下程序。

【实例 9.10】 创建一个帮助窗口，用来显示帮助信息。如图 9.14 所示。

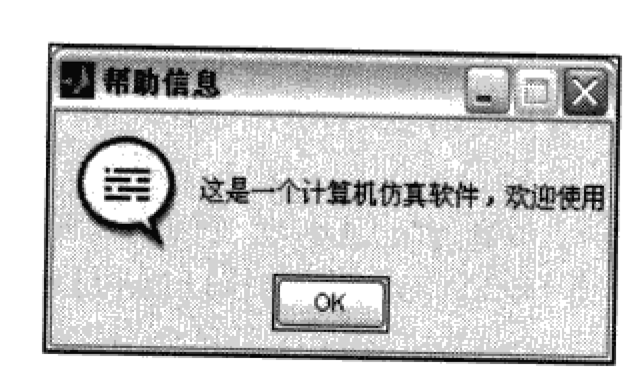


图 9.14　帮助文件建立

【实例讲解】 在程序代码中创建帮助窗口，可以增加了程序的友好性。

9.2.8 errordlg 函数——错误窗口对话框

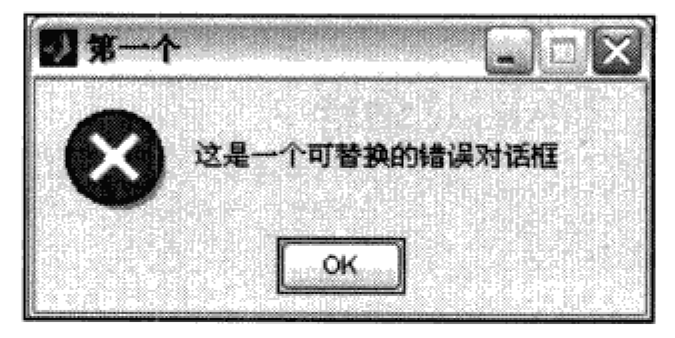
【语法说明】 H=errordlg(错误信息，标题，状态)。

【功能介绍】 此函数用来创建一个有标题的错误对话框，在对话框中显示错误信息。如果用户单击“OK”按钮，错误对话框消失，继续以下的程序。如果相同标题的对话框已经存在，并且状态处于“replace”时，将把原来的错误对话框替换掉。缺省值为“non-modal”，不替换原来的错误对话框。

【实例 9.11】 产生两个错误对话框。

```
>> errordlg('这是一个可替换的错误对话框','第一个','replace');
>>errordlg('这是一个不可替换的错误对话框','第二个',' non-modal');
```

通过上面两个命令得到的对话框如图 9.15 和图 9.16 所示。



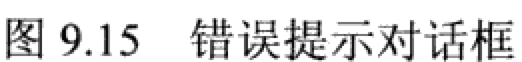

图 9.15　错误提示对话框



图 9.16　另外一个错误提示对话框

【实例讲解】 错误提示对话框是提示用户操作错误的重要手段。

9.2.9　warndlg 函数——警告对话框

【语法说明】 H=warndlg(警告信息,标题)

【功能介绍】 此函数用于创建一个具有标题的警告对话框，同时在警告对话框中显示警告信息。

【实例 9.12】 查看警告对话框。

```
>> warndlg('是否保存该信息？','保存信息')
```

得到的结果如图 9.17 所示。



图 9.17　警告对话框

【实例讲解】 警告对话框是程序的提示对话框，可以在用户进行重要操作之前，给予重要提示。

9.2.10　uisetcolor 函数——颜色设置对话框

【语法说明】

- C=uisetcolor(H,'对话框标题')。
- C=uisetcolor([r g b],'对话框标题')。

【功能介绍】 此函数将获得一个有自定义标题的颜色设置对话框，H 是图形对象的句柄，[r g b]是 RGB 向量。在对话框中选择合

适的颜色。

【实例 9.13】 建立一个用来设置颜色的对话框。

```
H=figure;
N=uisetcolor(H,'设置图片颜色');
```

得到的结果如图 9.18 所示。

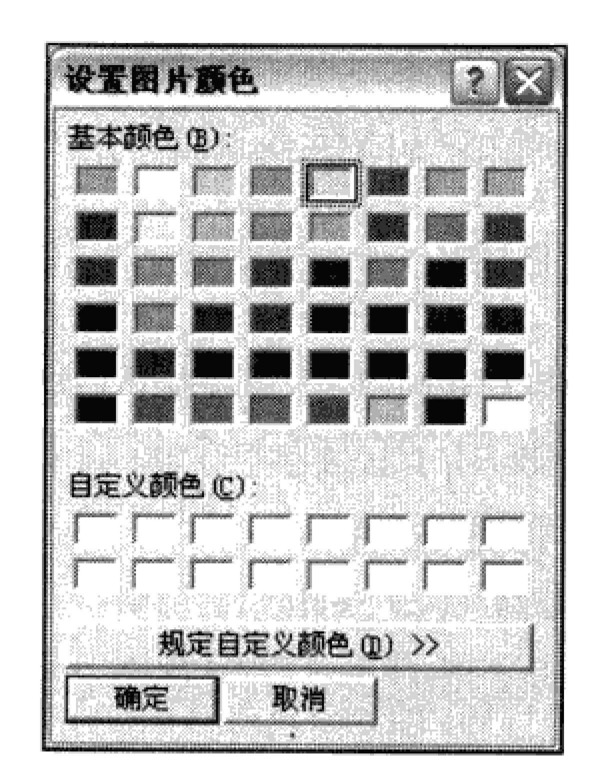


图 9.18 颜色设置对话框

【实例讲解】 用户可以在设置颜色的对话框中选择对应的颜色。

9.2.11 questdlg 函数——提问对话框设计

【语法说明】 H=questdlg(问题, 标题, 按钮名称 1, 按钮名称 2, 按钮名称 3, 缺省值)。

【功能介绍】 此函数用于创建一个具有标题的提问对话框，在对话框中将显示问题和 3 个按钮。按钮名称的返回值就是被选中的按钮名称，如果没有设定按钮名称，3 个按钮的名称就为缺省值，分别为 Yes、No、Cancel，缺省返回值为 Yes。

【实例 9.14】 设计提问式对话框。

```
>> A1=questdlg('是否保存该信息','信息保存')
>> A2=questdlg('是否保存该信息','信息保存','是','否','取消
','是');
```

得到的结果如图 9.19 和图 9.20 所示。




图 9.19　显示性对话框



图 9.20　另外类型的对话框

【实例讲解】 和警句对话框不同，提问对话框提供用户进行其他的选择。

9.2.12　msgbox 函数——消息框设计

【语法说明】 H=msgbox(消息内容, 标题, 图标类型)。

【功能介绍】 此函数用于创建一个具有标题的消息对话框，在对话框中将显示消息内容。

【实例 9.15】 创建一个具有标题的消息对话框。

```
>>msgbox('This is a message box','message','warn');
```

得到的结果如图 9.21 所示。



图 9.21　消息框

【实例讲解】 消息框广泛的应用于测试程序中。

9.2.13　uicontrol 函数——控件编写

【语法说明】 H=uicontrol('parent', 父控件句柄, 'style', '控件类型名', 'position', 位置坐标, 'callback', 回调函数)

【功能介绍】 此函数用于创建各种控件，控件的名称由控件类型决定。

【实例 9.16】 在空的图形界面上创建一个静态文本框。

```
    gcf=figure;
    set(gcf,'menubar','none')
    set(gcf,'unit','normalized','position',[0.2,0.2,0.64
,0.32]);
    set(gcf,'defaultuicontrolunits','normalized')
    uicontrol('style','frame',...
       'position',[0.67,0.55,0.25,0.25]);
    >> uicontrol('style','text',...       %建立一个静态文本框
       'string','文本框:',...
       'position',[0.68,0.77,0.18,0.1],...
       'horizontal','left');
         hr1=uicontrol(gcf,'style','radio',...  %  创建选择
控件
       'string','正体',...
       'position',[0.7,0.69,0.9,0.08]);
   set(hr1,'value',get(hr1,'Max'));
   set(hr1,'callback',[...
                  'set(hr1,''value'',get(hr1,
''max'')),',...
                  'set(hr2,''value'',get(hr2,
''min'')),',...
                  'set(htitle,''fontangle'',
''normal''),',...
              ]);
   hr2=uicontrol(gcf,'style','radio',...    %     创建另外
一个选择控件
              'string','斜体',...
              'position',[0.7,0.58,0.9,0.08],...
              'callback',[...
                  'set(hr1,''value'',get(hr1,
''min'')),',...
                  'set(hr2,''value'',get(hr2,
''max'')),',...
                  'set(htitle,''fontangle'',
''italic'')',...
              ]);
```

得到的结果如图 9.22 所示。

【实例讲解】 除了使用程序代码之外，用户还可以直接向窗体中添加控件。

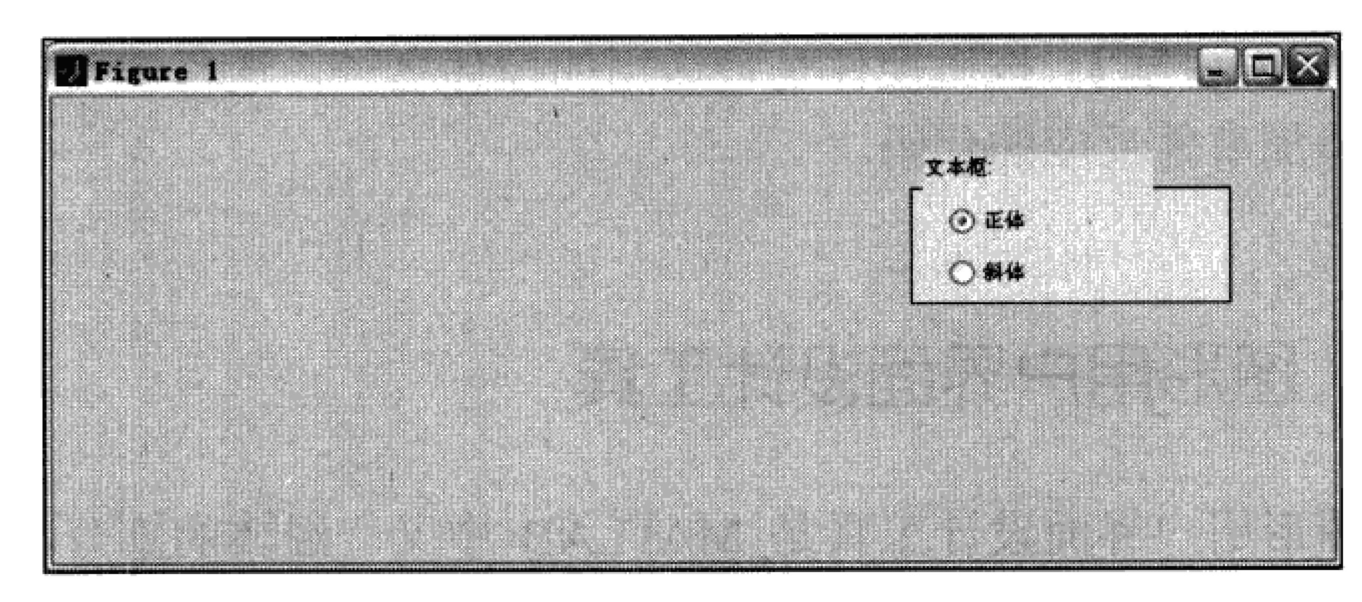


图 9.22 文本框和单选控件的创建

9.2.14 Button 按钮控件的设计

【实例 9.17】 制作一个能绘制任意图形的交互界面。它包括可编辑文本框、弹出框、列表框。本例的关键内容是：如何使编辑框允许输入多行指令。

```
>> gcf=figure;
set(gcf,'menubar','none')
set(gcf,'unit','normalized','position',[0.2,0.2,0.3,
0.3]);
set(gcf,'defaultuicontrolunits','normalized')
hpush=uicontrol(gcf,'Style','push',...
   'position',[0.2,0.2,0.16,0.13],'string','Apply');
```

得到的图形如图 9.23 所示。

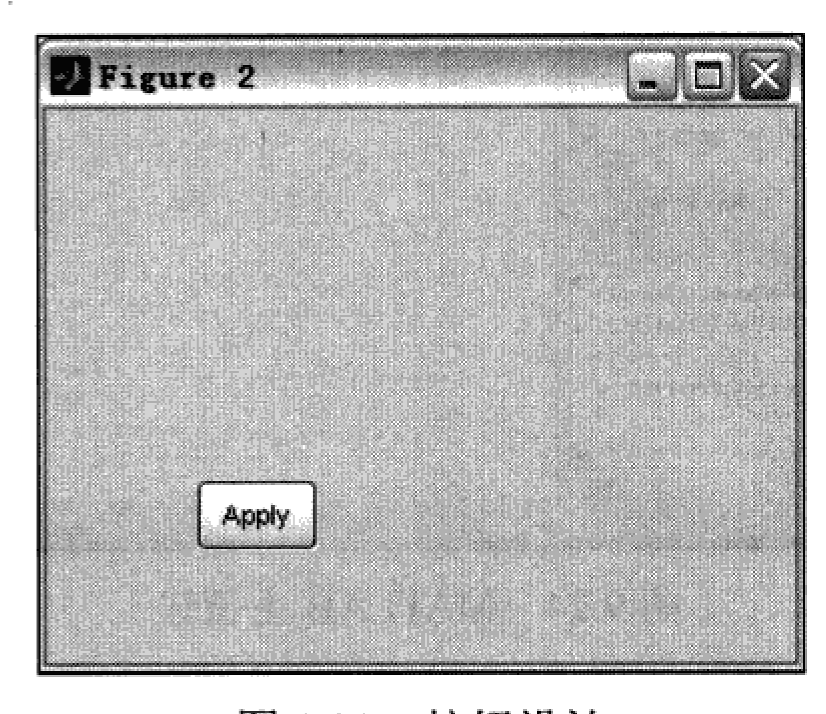


图 9.23 按钮设计

【实例讲解】 按钮控件是最通用的控件之一，通常需要为按钮控件添加事件反应的代码。

9.3 图形用户界面设计工具

图形用户界面设计工具是 MATLAB 中又一经典的部分，在上一节中讲解了几种常见控件和菜单项的函数实现。利用编写程序手工实现界面设计的一个优点就是灵活，但是，即使是一个小控件的生成就需要编写繁琐的程序，这无疑增加了图形用户界面的开发时间，MATLAB 自身提供的图形用户界面设计工具很好地解决了这一难题。本节将要讲解图形用户界面设计工具的相关知识，并通过实例讲解具体用法。

9.3.1 界面设计工具的结构

在 MATLAB 7.0 的主窗口中，单击如图 9.24 所示的“GUIDE”按钮。

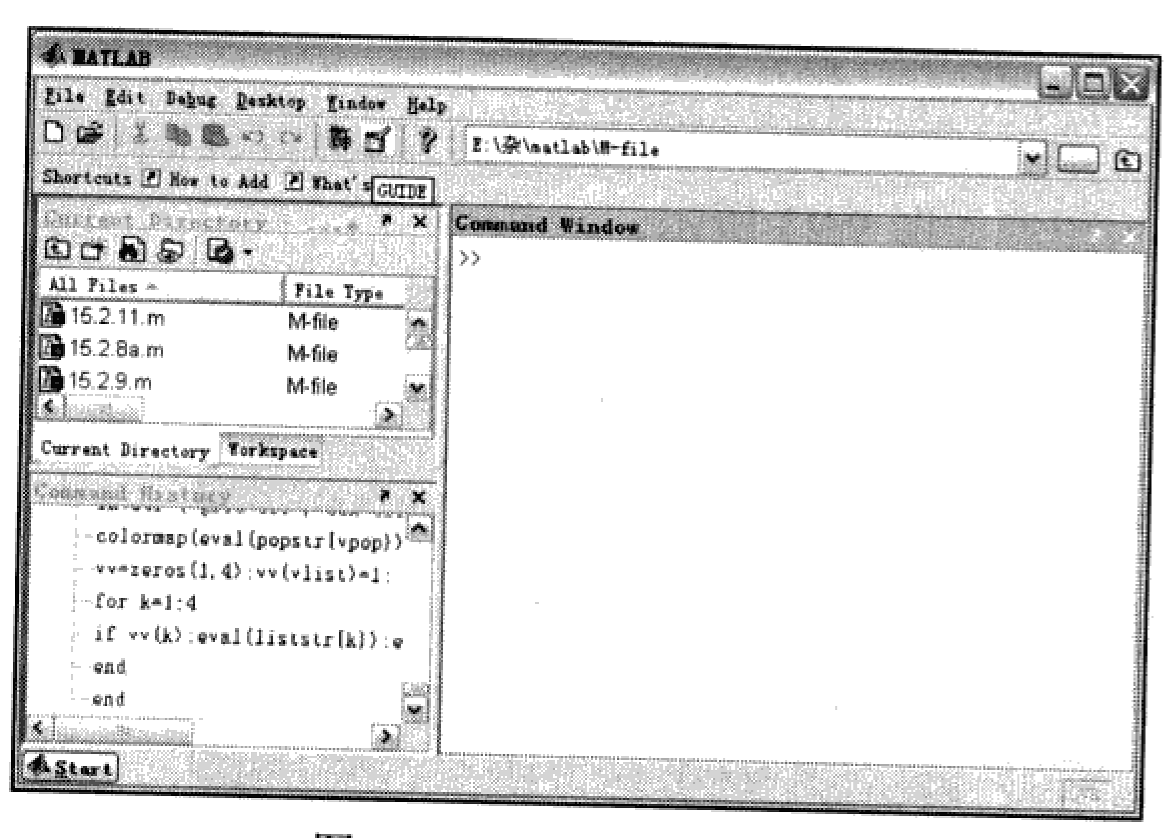


图 9.24 MATLAB 主窗口

弹出图形用户界面设计工具的向导界面，如图 9.25 所示。在这

个向导中有两个 Page 页，第一个用来新建一个 GUI，共有 4 种模版可供选择，分别是：

- 空白窗口；
- 带有一些控制件的模版；
- 带有菜单栏的模版；
- 带有对话框的模版。

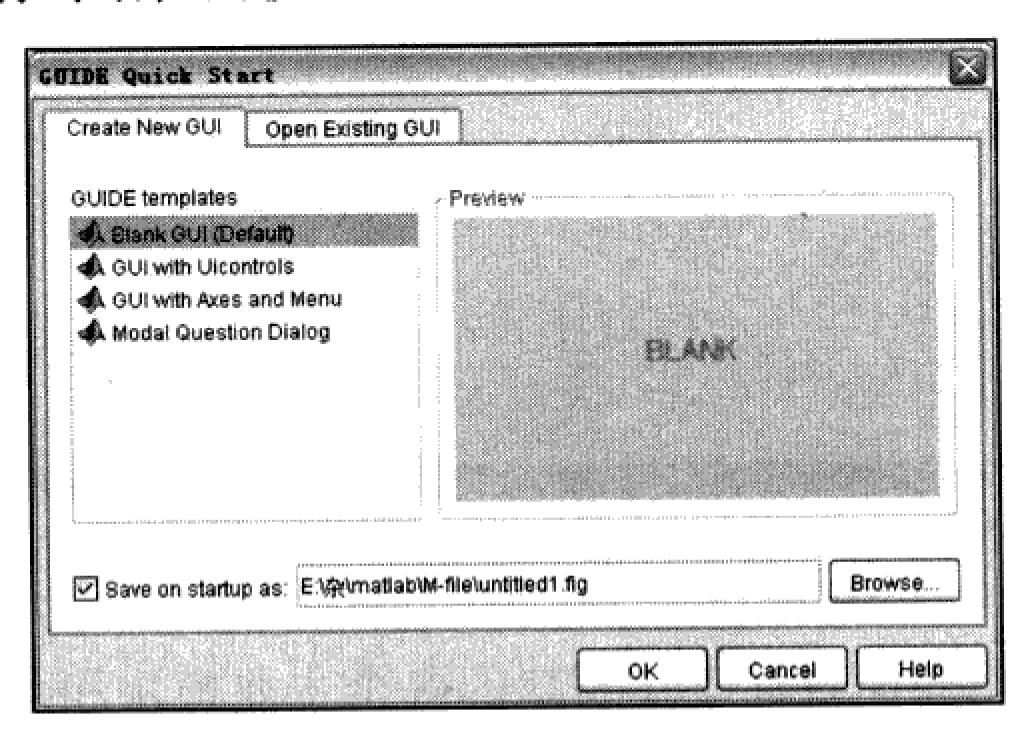


图 9.25 GUIDE 初始界面

第二个 Page 页用来打开已经存在的 GUI，如图 9.26 所示，MATLAB 中提供了显示最近打开过的文件功能，在这些文件中可以方便地打开需要的那个。

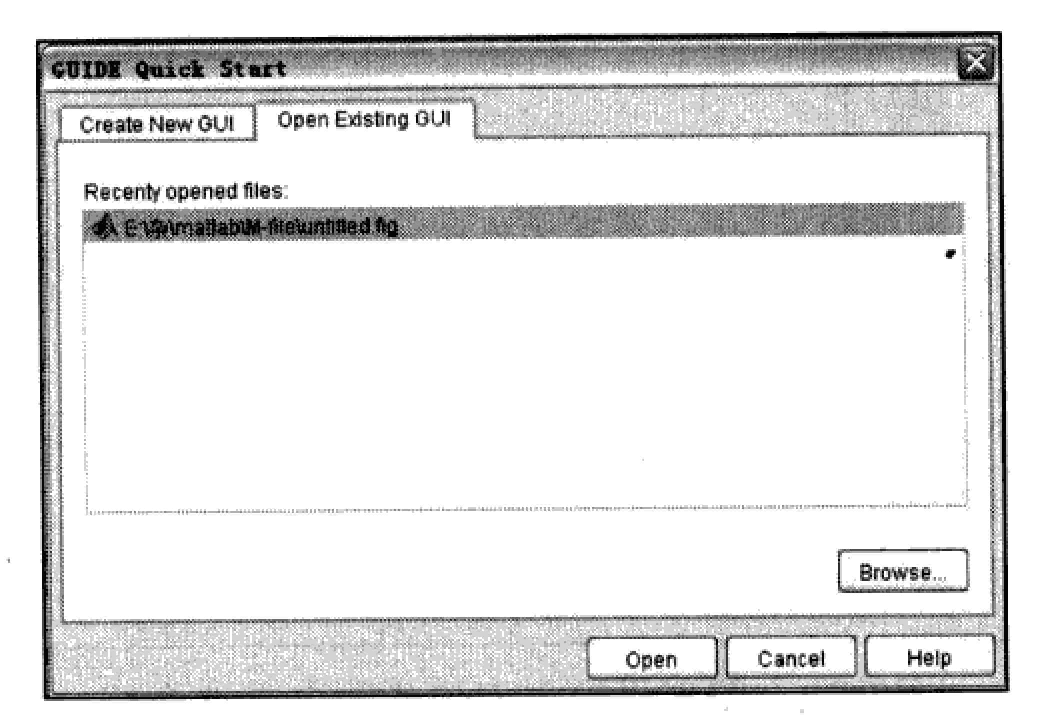


图 9.26 打开已经存在的 GUI

图形用户界面设计的工作区如图 9.27 所示。

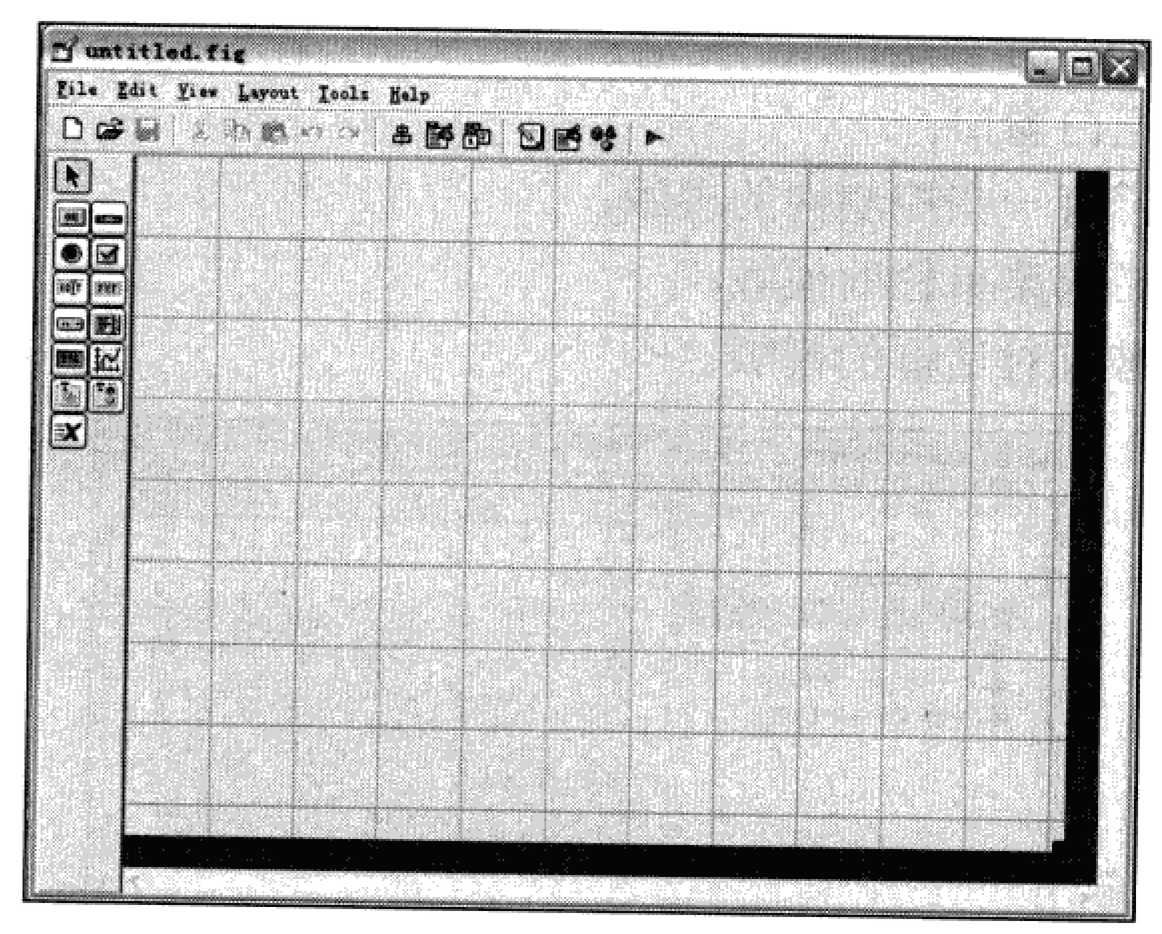


图 9.27　图形设计界面工作区

图形用户界面设计工作窗口的各个主要模块和菜单的功能如图 9.28 所示。

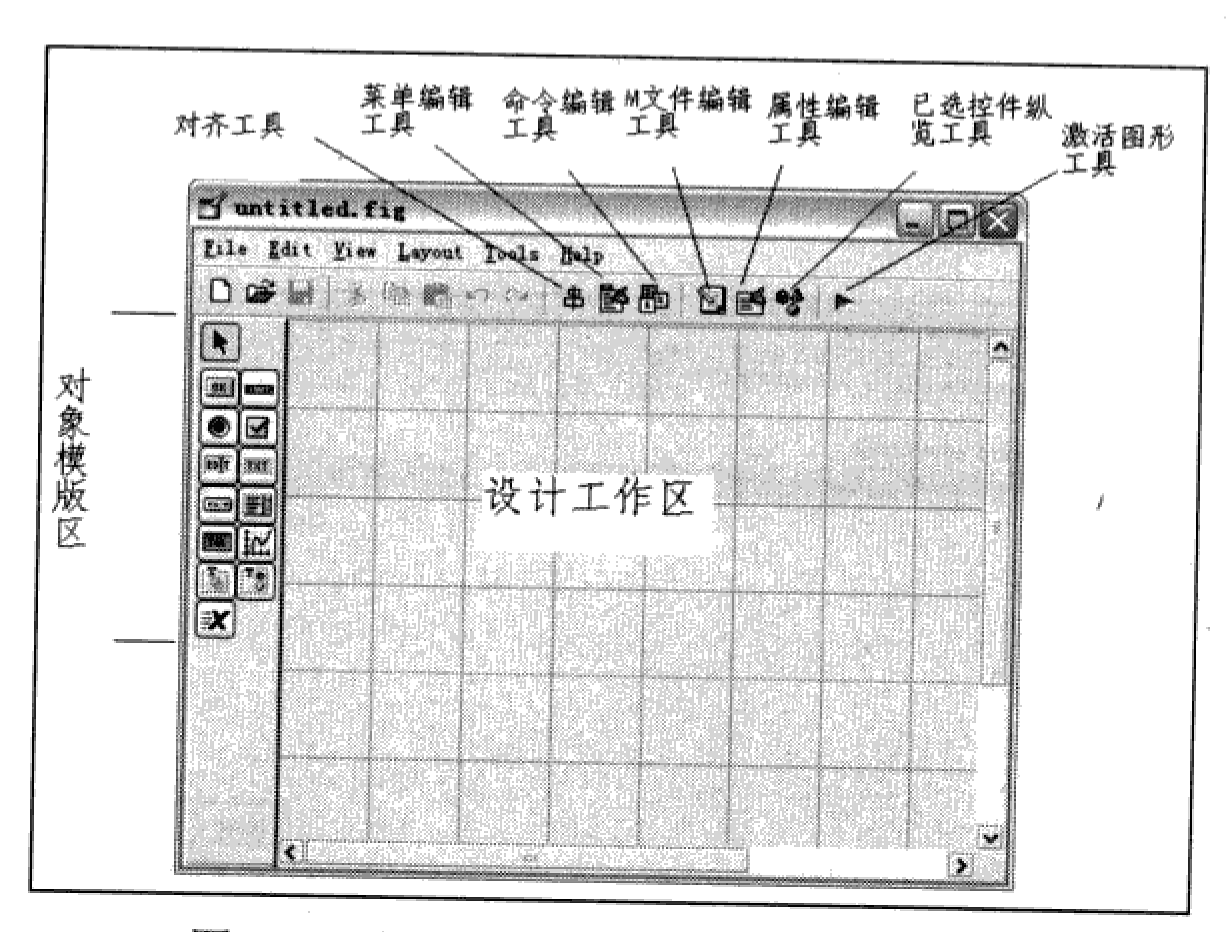


图 9.28　图形设计界面工作区各功能介绍

9.3.2 用户界面设计工具的控件介绍

在对象模版区中有很多控件可供选用，下面从最简单的 Push Button 控件为例作讲解。

Push Button 控件是 MATLAB 中最常用的控件，首先将这个控件从模版区中拖放到 GUI 设计工作区，双击控件体，弹出如图 9.29 所示的属性编辑窗口，在这个窗口中包括了所有用于设计 Button 控件的属性，可以对所有或部分的属性进行设置。

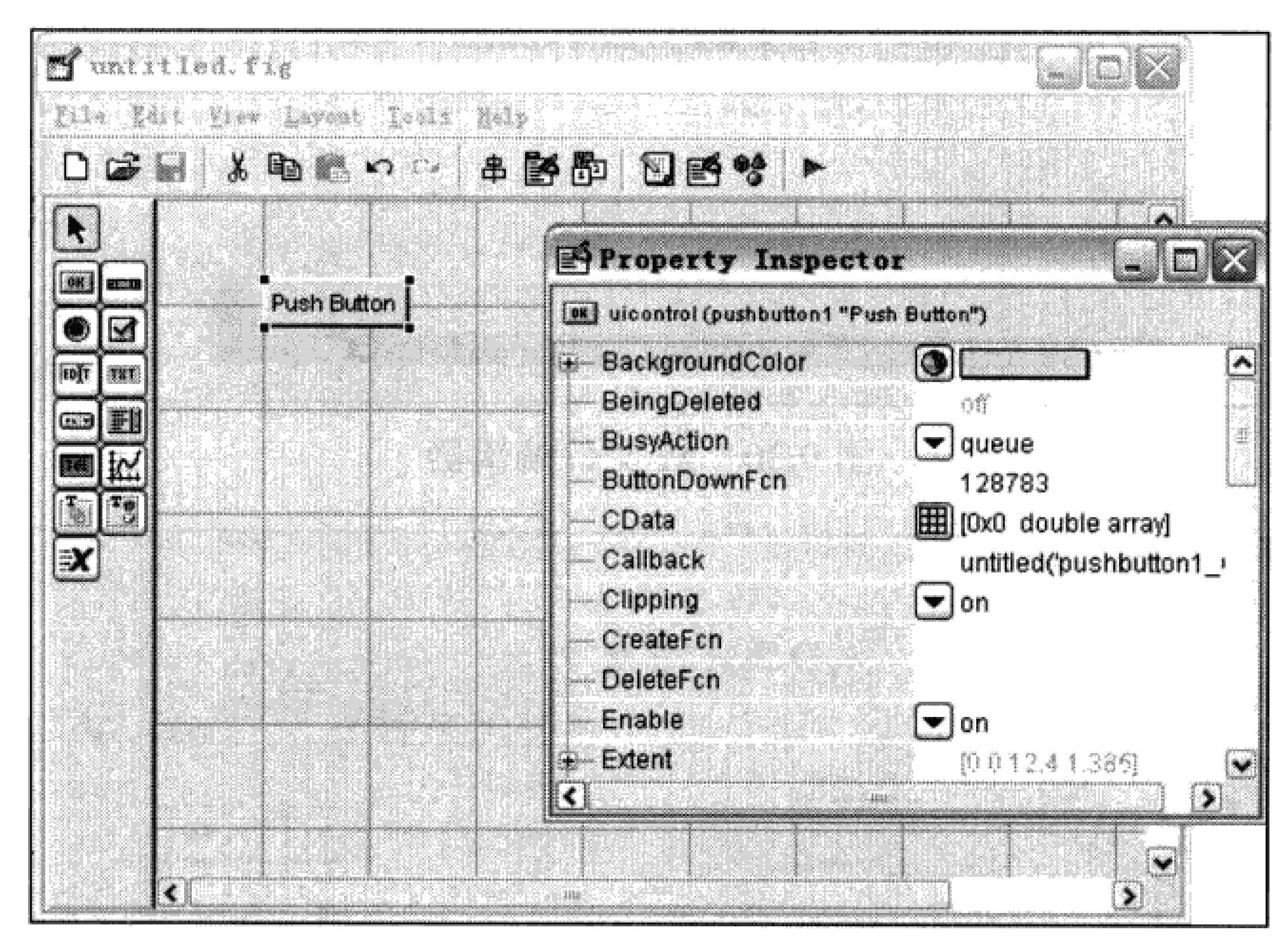


图 9.29 属性编辑器

每个控件都有自己的属性编辑器，如果要使上面的 Button 控件运行的时候产生反应，要选择菜单栏中的“M 文件编辑器”，如图 9.30 所示，将需要操作的内容写在 M 文件中。

其他控件的应用情况与此相同，如果在一个界面上使用多个控件，会用到界面布局编辑器，如图 9.31 所示。

选中需要对齐的控件，然后选择一种对齐方式，并单击“OK”按钮即可。如图 9.32 所示选择了左对齐方式，所以所选控件全部按

左对齐方式对齐。

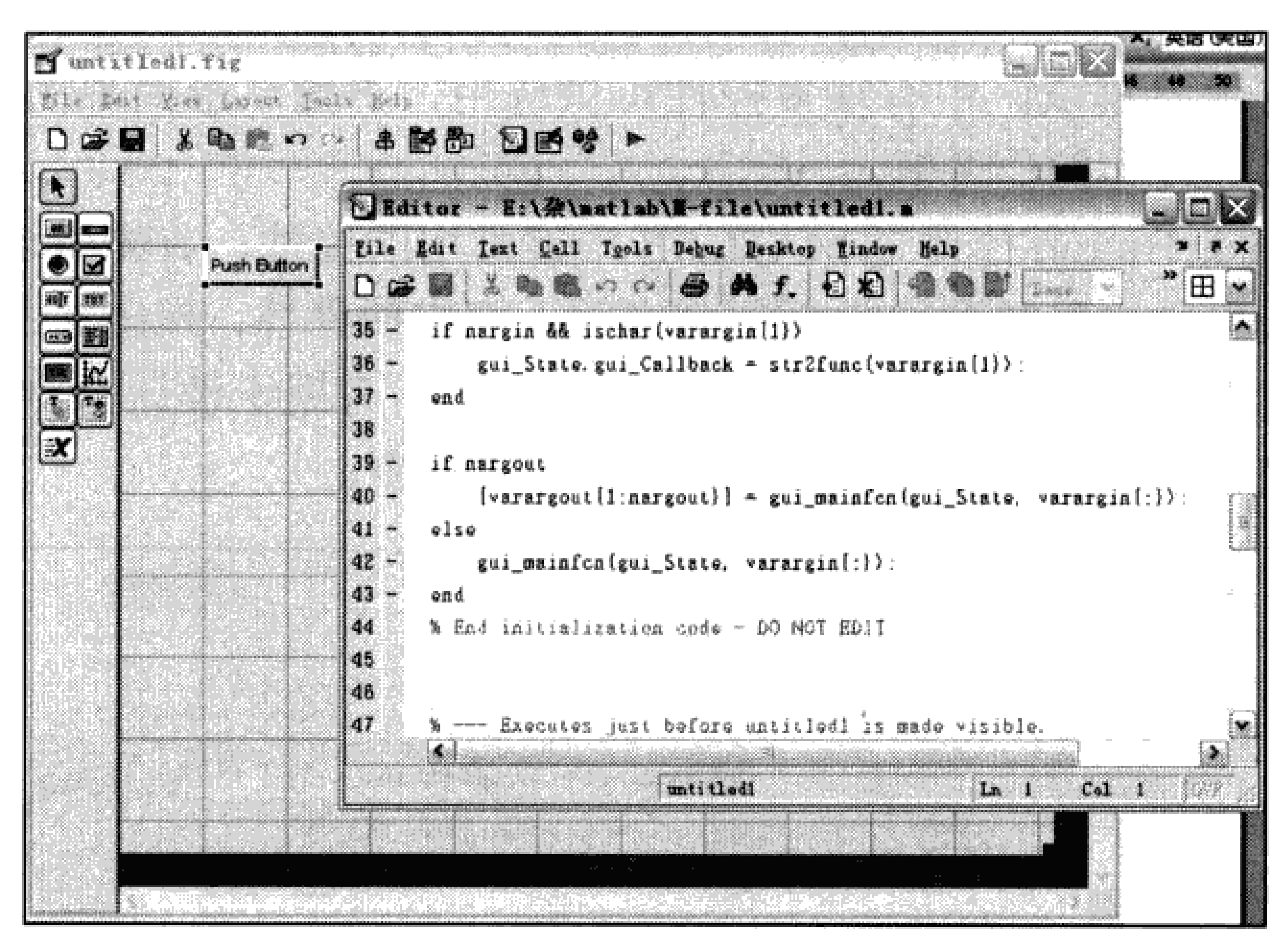


图 9.30　M 文件编辑器

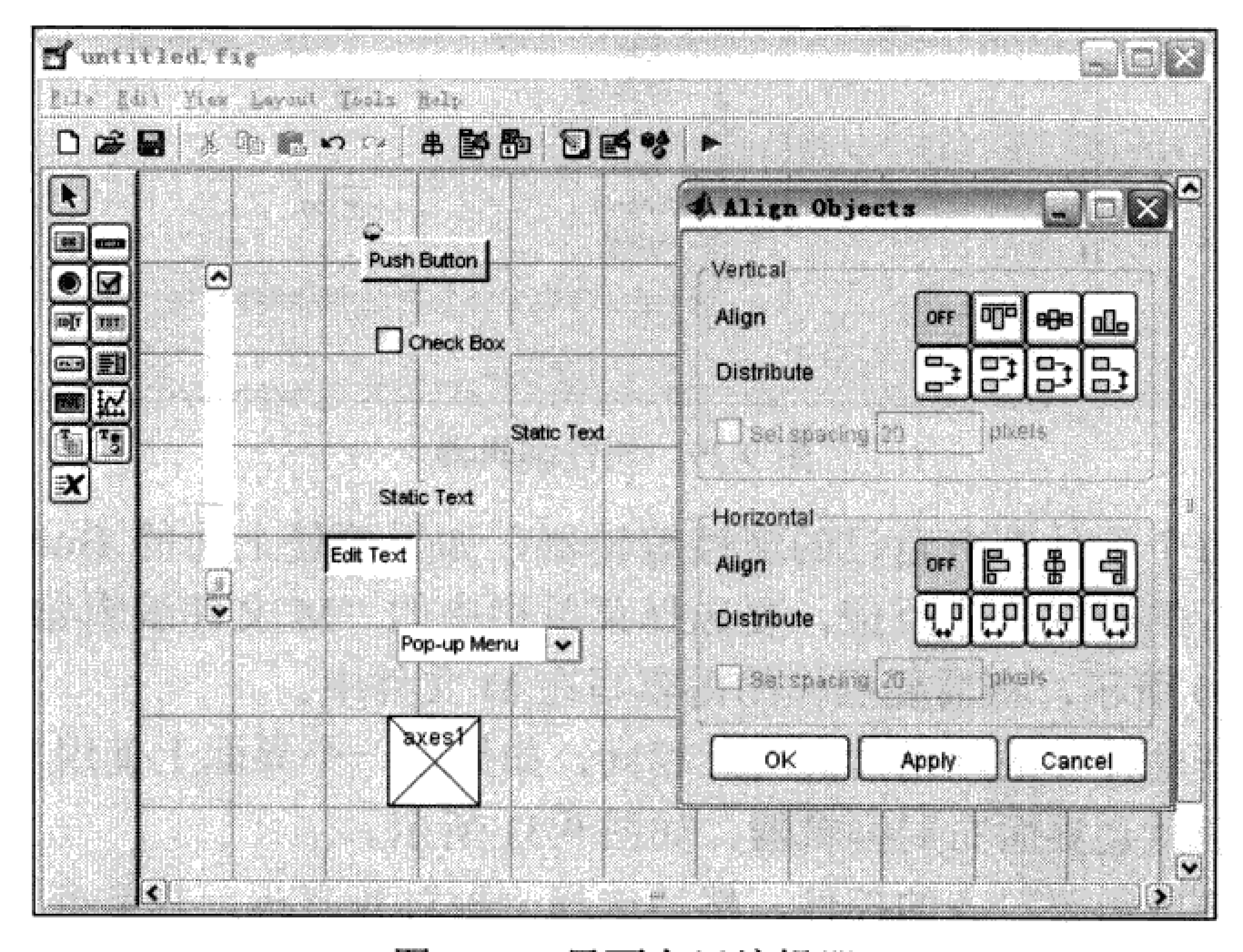


图 9.31　界面布局编辑器

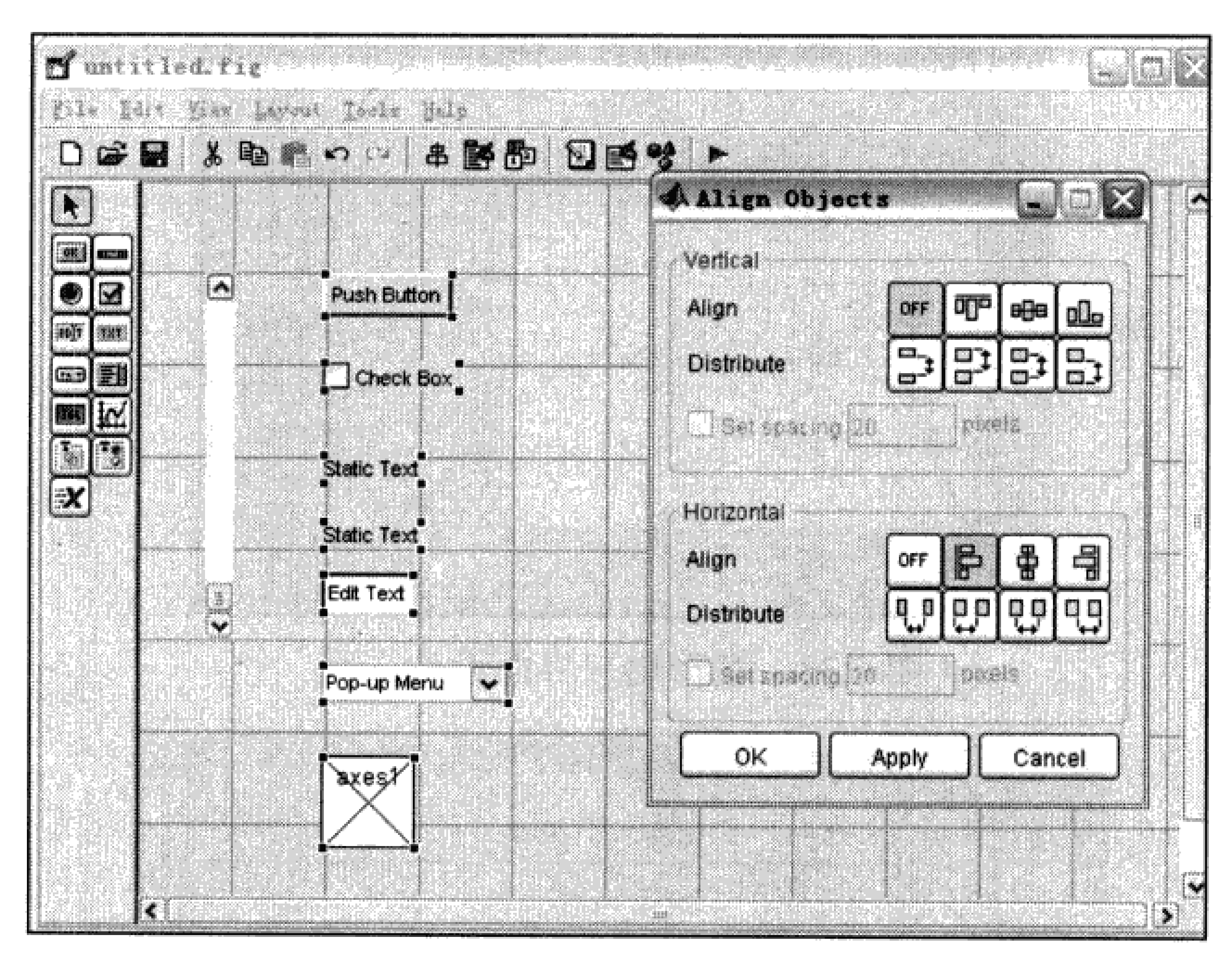


图 9.32　控件左对齐

9.3.3　交互式用户界面设计工具应用示例

【实例 9.18】 使用 guide 来创建一个如图 9.33 所示的图形用户界面。该界面具有如下功能。

在编辑框中，可输入表示阻尼比的标量或“行数组”数值，并在按【Enter】键后，在轴上画出相应的蓝色曲线。坐标范围：*X* 轴 [0, 9]；*Y* 轴 [0, 2]。

在单击【Grid on】或【Grid off】键时，在轴上画出或删除“分格线”，缺省时，无分格线。

在菜单【Options】下，有两个下拉菜单项【Box on】和【Box off】，缺省时为 Box off 状态。

所设计的界面和其上图形对象、控件对象都按比例缩放，如图 9.33 所示。

（1）选定需要的控件，并放到设计工作区中。如图 9.34 所示。

（2）为每个控件设置属性，如图 9.35 所示。

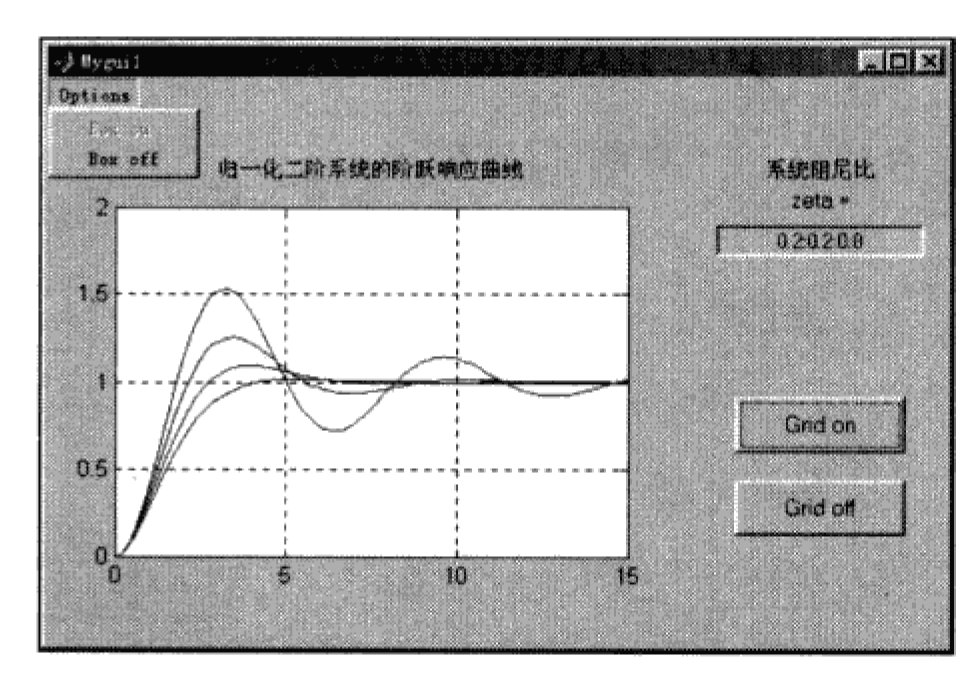


图 9.33　图形用户界面设计实例

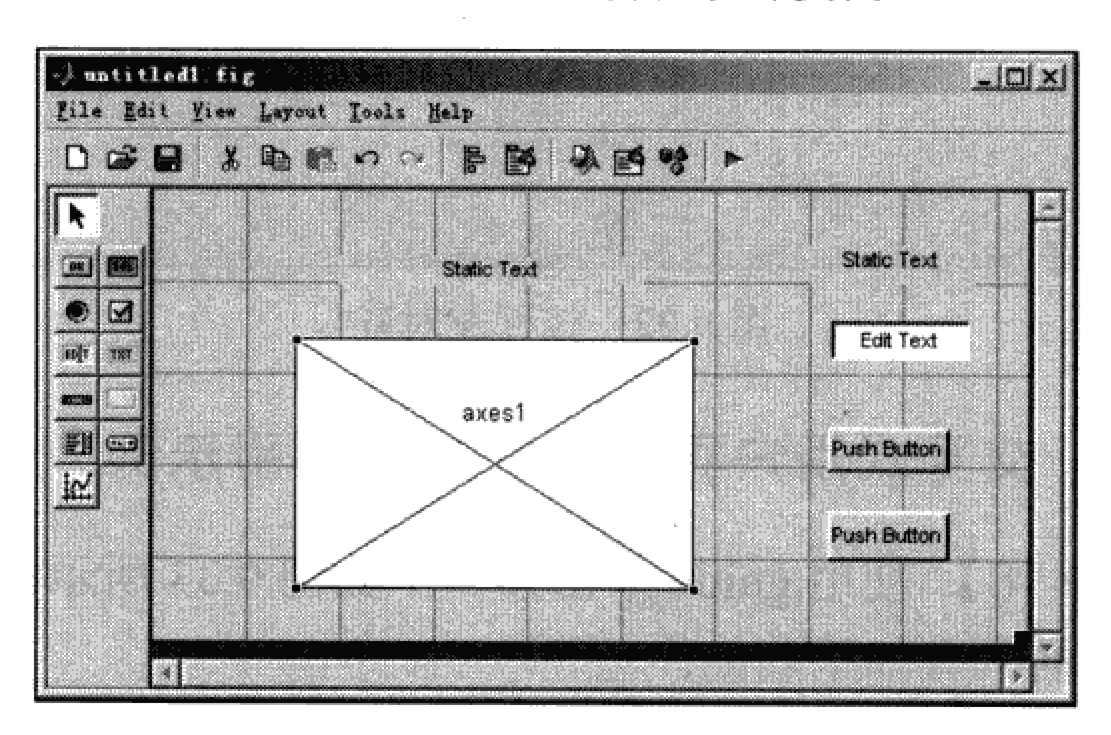


图 9.34　控件拖放

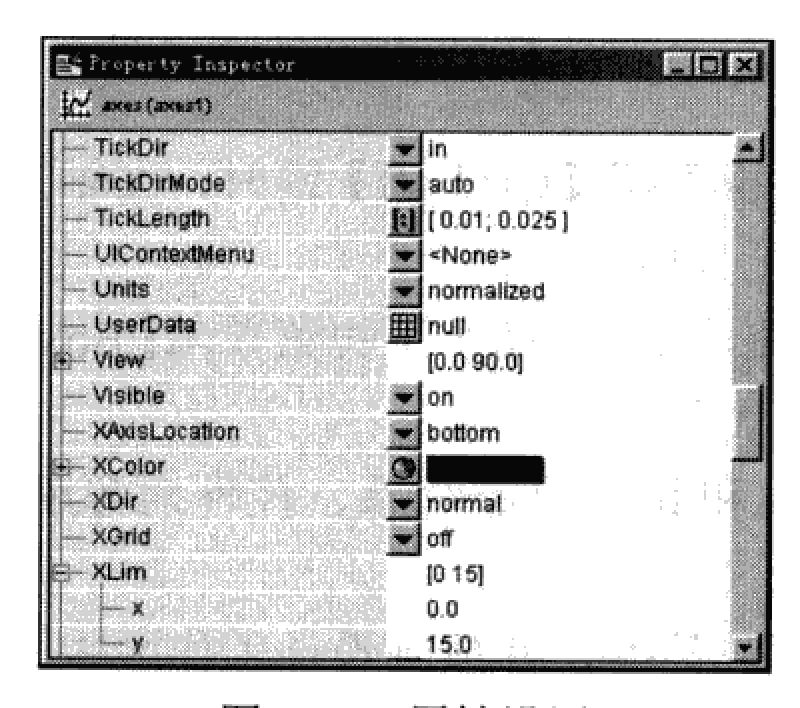


图 9.35　属性设置

（3）规划布局界面，如图 9.36 所示。

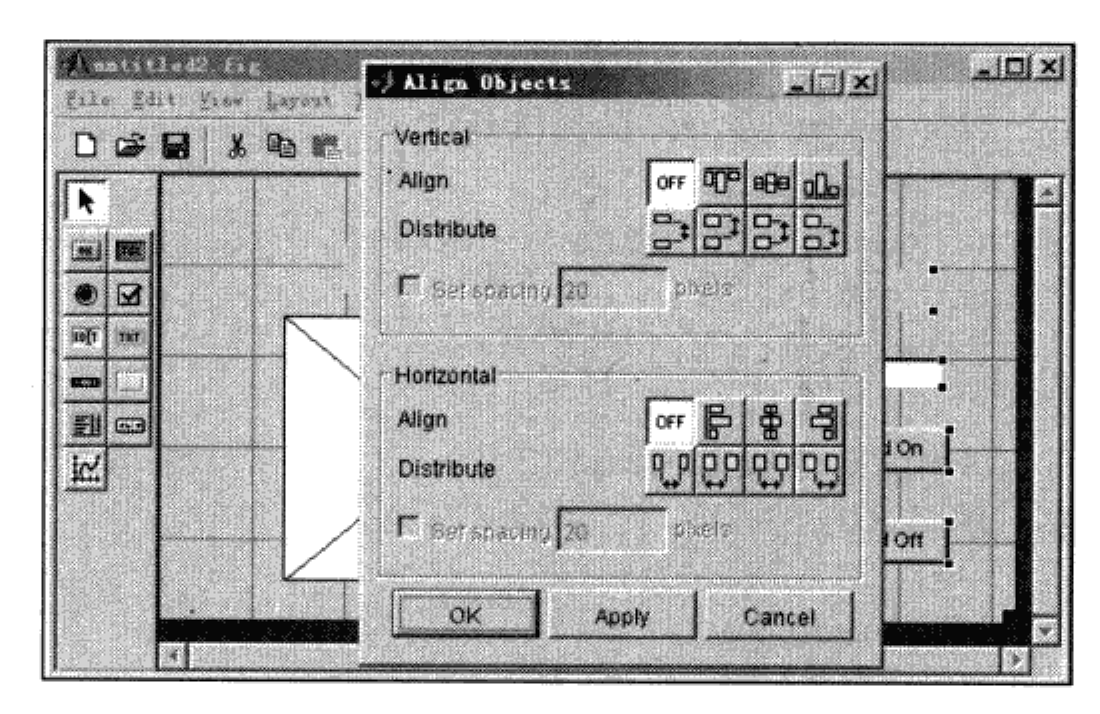


图 9.36 布局方式编辑

（4）设置菜单编辑器，如图 9.37 所示。

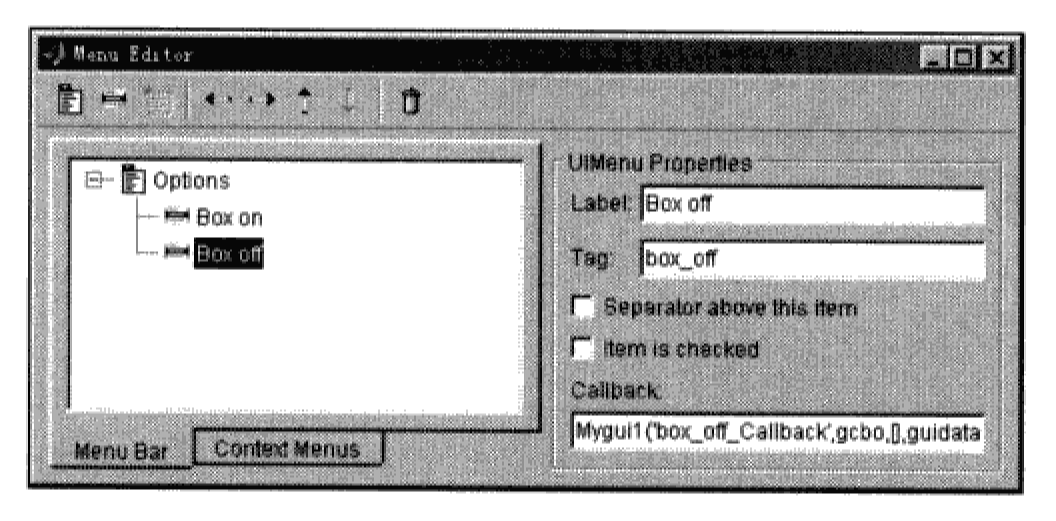


图 9.37 菜单编辑器

（5）在 M 文件编辑器中添加内容。在 zeta_edit_Callback 和 box_off_Callback 函数中添加如下内容：

```
    function varargout = Mygui1(varargin)
    % MYGUI1 Application M-file for Mygui1.fig
    %    FIG = MYGUI1 launch Mygui1 GUI.
    %      MYGUI1('callback_name', ...) invoke the named
callback.
    % Last Modified by GUIDE v2.0 9-Jun-2002 16:12:52
    if nargin == 0  % LAUNCH GUI
        fig = openfig(mfilename,'reuse');
        % Use system color scheme for figure:
        set(fig,'Color',get(0,'defaultUicontrolBackground
```

```
Color'));
        % Generate a structure of handles to pass to
callbacks, and store it.
        handles = guihandles(fig);
        guidata(fig, handles);
        set(handles.box_off,'enable','off')
        if nargout > 0
            varargout{1} = fig;
        end
    elseif ischar(varargin{1})          % INVOKE NAMED
SUBFUNCTION OR CALLBACK
        try
            if (nargout)
                [varargout{1:nargout}] = feval(varargin
{:}); % FEVAL switchyard
            else
                feval(varargin{:}); % FEVAL switchyard
            end
        catch
            disp(lasterr);
        end

    end

    % --------------------------------------------------------
    function varargout = GridOff_push_Callback(h, eventdata,
handles, varargin)
    grid off
    % --------------------------------------------------------
    function varargout = GridOn_push_Callback(h, eventdata,
handles, varargin)
    grid on
    %--------------------------------------------------------
    function varargout = zeta_edit_Callback(h, eventdata,
handles, varargin)
    z=str2num(get(handles.zeta_edit,'String'));
    t=0:0.1:9;
    cla
    for k=1:length(z)
       y(:,k)=step(1,[1, 2*z(k), 1], t);
       line(t,y(:,k));
```

```
    end
    % ------------------------------------------------------------
    function varargout = options_Callback(h, eventdata, 
handles, varargin)
    % ------------------------------------------------------------
    function varargout = box_on_Callback(h, eventdata, 
handles, varargin)
    box on
    set(handles.box_on,'enable','off')
    set(handles.box_off,'enable','on')
    % ------------------------------------------------------------
    function varargout = box_off_Callback(h, eventdata, 
handles, varargin)
    box off
    set(handles.box_off,'enable','off')
    set(handles.box_on,'enable','on')
```

（6）运行设计完成的用户界面，如图 9.38 所示。

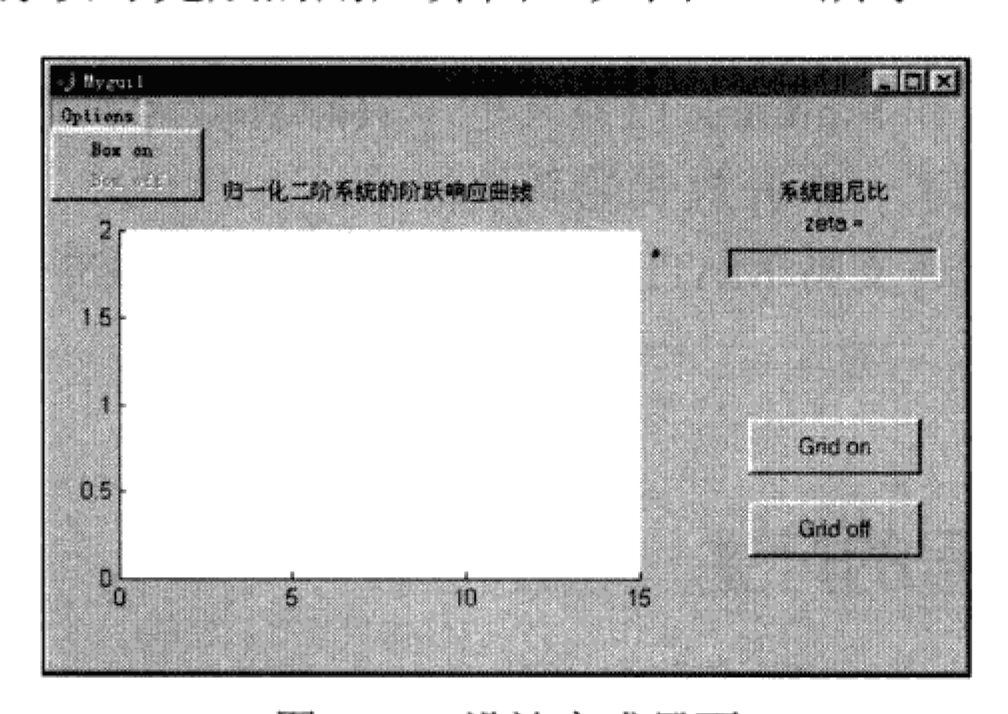
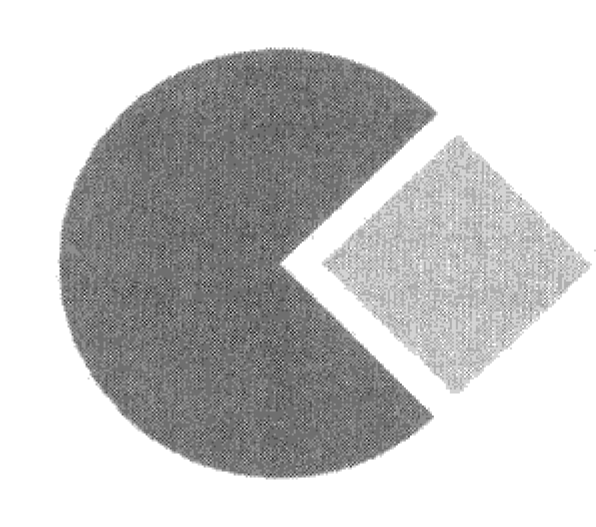


图 9.38　设计完成界面

【实例讲解】 用户可以将图形用户界面和程序代码的设置方法对比。



第 10 章　信号处理工具箱

工具箱是 MATLAB 提供的用于解决特定领域问题的 M 文件的集合。MATLAB 包含很多个工具箱，每个工具箱都又包含丰富的函数和工具。这里只选取几个工程上最常用的工具箱加以介绍，本章将详细介绍信号处理工具箱，它的主要特点是：

- 具有稳定的数值特性和很高的精度；
- 可以和 Simulink 以及其他工具箱进行天衣无缝的结合；
- 组成了一个开放式的系统；
- 在所有支持 MATLAB 的平台上都是通用的，因此有极好的可移植性。

10.1　信号的产生

信号是信号处理最基本的概念，所有的信号处理工具箱都是围绕着信号展开的。所以，如何产生信号是最基础也是最重要的内容，本节就要介绍信号的产生问题。

10.1.1　三角信号产生

【功能介绍】 生成单位冲击信号。

【实例 10.1】 生成长度为 6 点的三角信号。

```
>>  t=1:1:6

t =

     1     2     3     4     5     6

>> A=10

A =

    10
>> w=2*pi

w =

    6.2832

>>  signal=A*sin(w*t)

signal =

   1.0e-013 *

   -0.0245  -0.0490  -0.0735  -0.0980  -0.1225  -0.1470
```

实例运行效果如图 10.1 所示。

【实例讲解】 在 signal=A*sin(w*t)命令中，A 表示幅值，为 10，w 为周期，是 2*pi,pi 为圆周率，t 为 1～6 的离散时间点。最后得到的结果如图 10.1 所示。

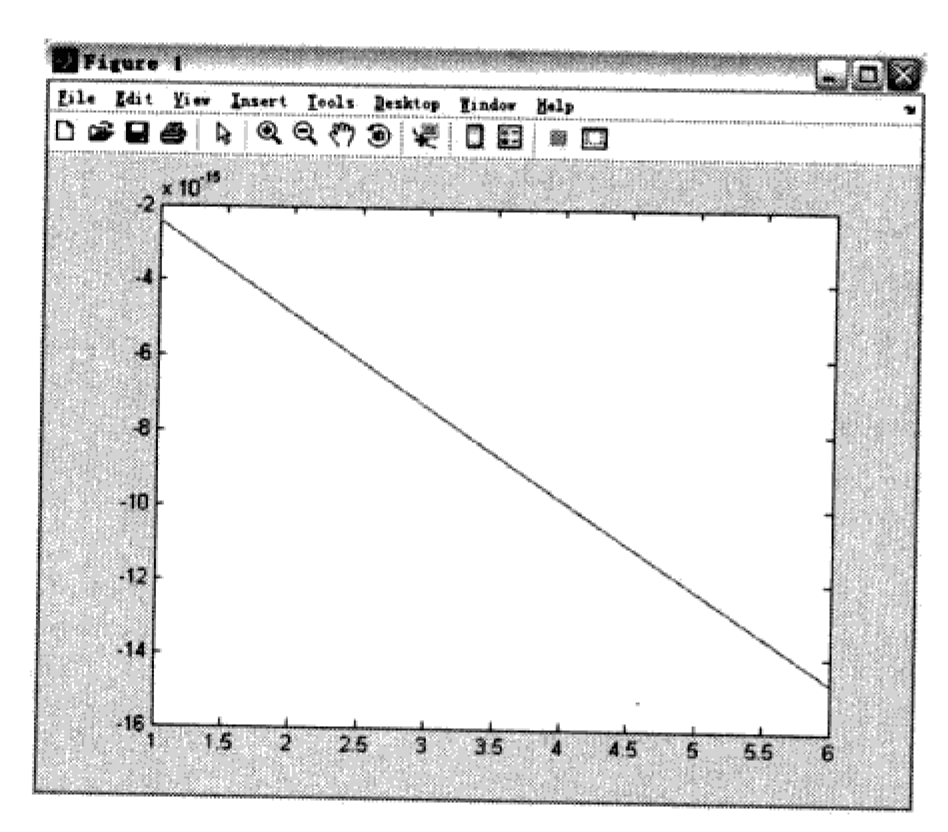


图 10.1　三角信号

10.1.2　ones 函数——单位阶跃信号的产生

【功能介绍】 signal=ones(1,n) ：n 表示长度。

【功能介绍】 生成单位阶跃信号。

【实例 10.2】 生成长度为 6 点的阶跃信号。

```
>> n=6

n =

     6

>> signal=ones(1,n)

signal =

     1     1     1     1     1     1
```

【实例讲解】 在默认的情况下，单位阶跃信号的幅值为 1。

10.1.3　单位冲击信号的产生

【功能介绍】 signal=[1，zeros(1,n−1)] ：n 表示长度。

【功能介绍】 生成单位冲击信号。

【实例 10.3】 生成长度为 6 点的单位冲击信号。

```
>> n=6

n =

     6

>> signal=[1,zeros(1,n-1)]

signal =

     1     0     0     0     0     0
```

【实例讲解】 在信号工具箱中，signal 是一个重要的关键字。

10.1.4 diric 函数——生成狄里克力函数

【背景知识】 狄里克力函数是工程中用得较多的函数，它的定义为：

$$diric(x)=\begin{cases}(-1)^{k(n-1)}, x=2\pi k, k=0,\pm1,\pm2...\\ \dfrac{\sin(nx/2)}{n\sin(x/2)},\text{其他情况}\end{cases}$$

【语法说明】 diric(x,n) ：x 为一个参考信号。

【功能介绍】 生成狄里克力函数。

【实例 10.4】 产生一个以 linspace 函数为参考信号的狄里克力函数。

```
>> x=linspace(0,4*pi,300);
>> plot(x,diric(x,9))
```

得到的图形如图 10.2 所示。

【实例讲解】 diric 函数会根据第一个参数的长度决定产生信号的长度。

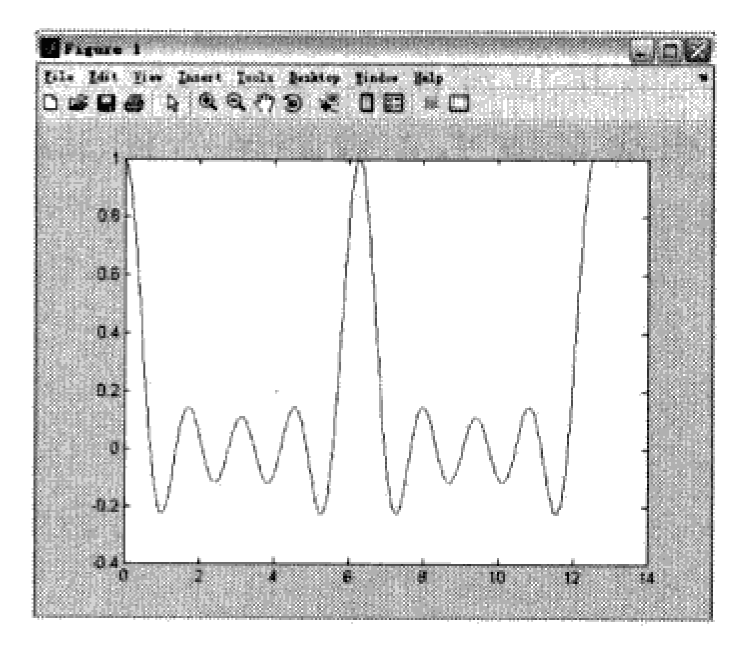


图 10.2　狄里克力函数

10.1.5　sawtooth 函数——生成锯齿波

【语法说明】 sawtooth(t，width)：width 参数指定锯齿波的波峰出现的位置，width=1 为正极性，width=0 为负极性，width=0.5 为对称锯齿波。

【功能介绍】 生成锯齿波。

【实例 10.5】 分别生成正极性、负极性和对称锯齿波。

```
>> t=1:2:20;
>> plot(t,sawtooth(t,1))
>> plot(t,sawtooth(t,0))
>> plot(t,sawtooth(t,0.5))
```

得到的锯齿波的图形如图 10.3～图 10.5 所示。

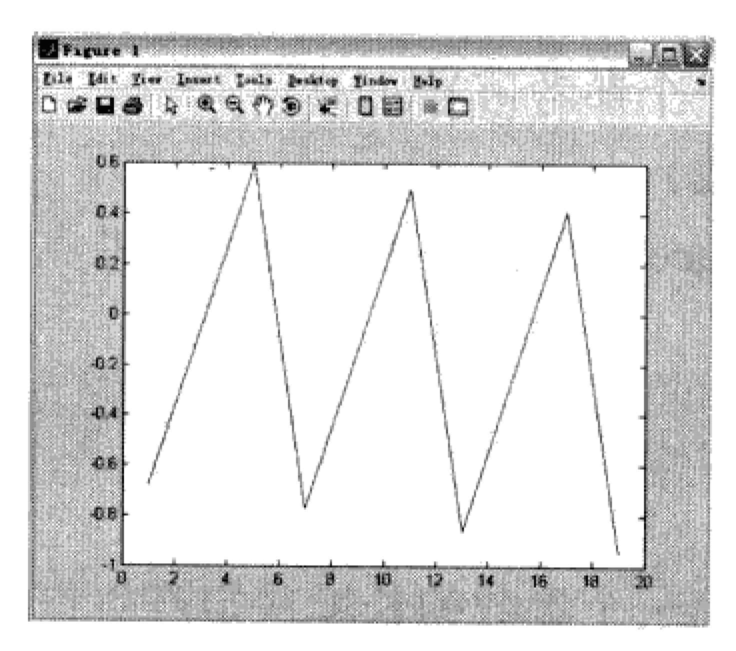


图 10.3　正极性锯齿波

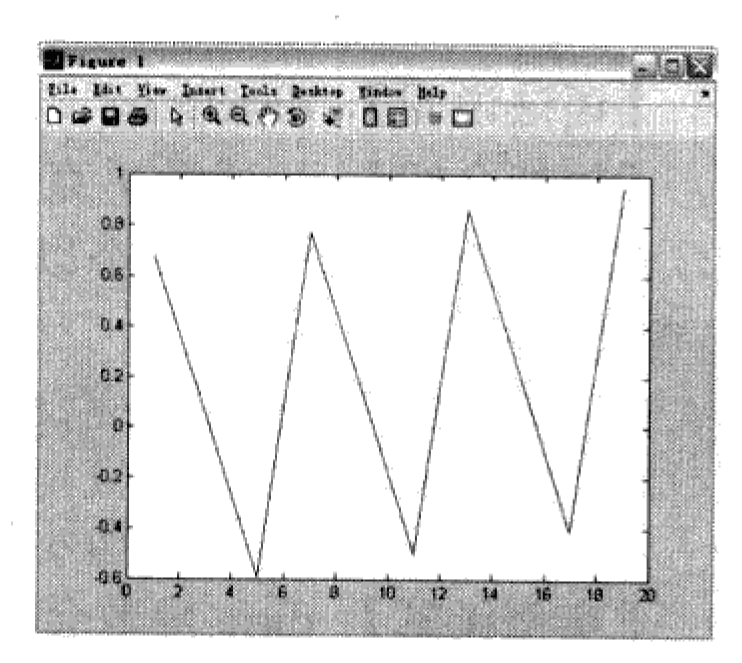


图 10.4　负极性锯齿波

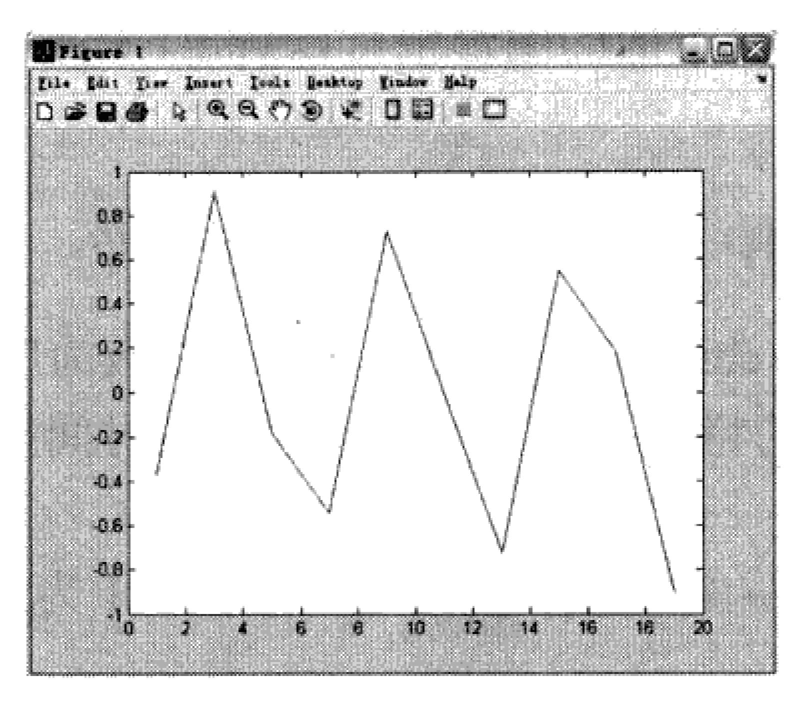


图 10.5　对称锯齿波

【实例讲解】 锯齿波是十分常用的信号类型。在信号工具箱中，用户可以使用 sawtooth 很便利地产生各种类型以及组合锯齿波。

10.1.6　sinc 函数——生成 *sinc* 信号

【语法说明】 sinc(t)。

【功能介绍】 产生 *sinc* 信号。

【实例 10.6】 产生 *sinc* 信号。

```
>>t=1:2:20;
>> plot(t,sinc(t))
```

得到的结果如图 10.6 所示。

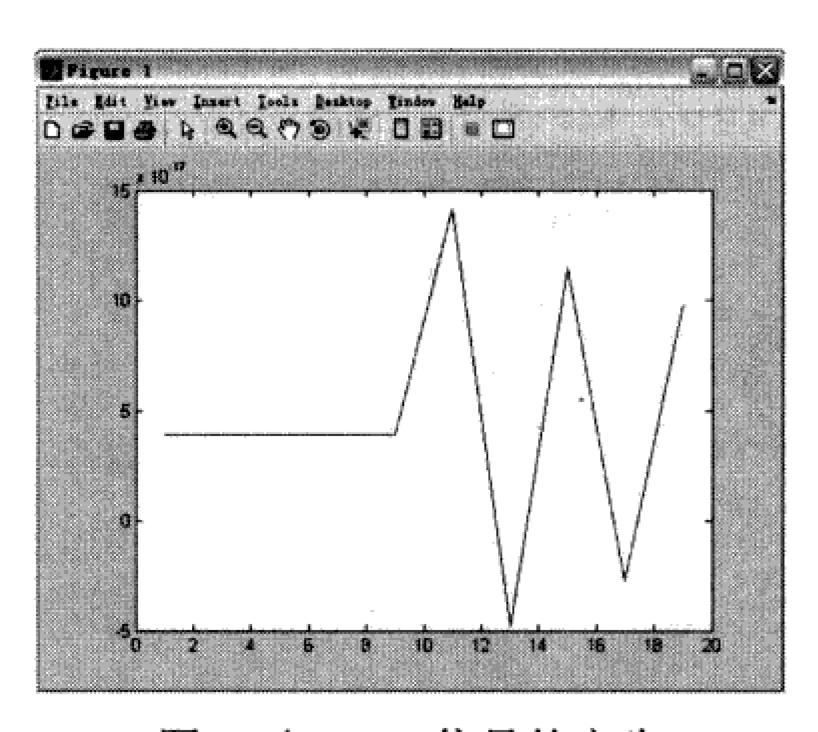


图 10.6　*sinc* 信号的产生

【实例讲解】 从上面的图形中可以看出，*sinc* 信号的形状和 sin 很类似。

10.1.7 chirp 函数——生成扫频信号

【背景知识】 chirp 信号在雷达中应用得较为广泛，又称为扫频信号，chirp 信号的特点是信号的瞬时频率随时间按一定规律变化。

【语法说明】

- Y=chirp(t，f0,t1,f1)：f0 为起始频率，t1 为开始变频的时刻，f1 为频率的增长值。
- Y=chirp(t，f0,t1,f1,'method')：f0 为起始频率，t1 为开始变频的时刻，f1 为频率的增长值，method 决定信号的瞬时频率，有 3 种形式可以选择，分别是 linear、quadratic 和 logrithmic。
- Y=chirp(t，f0,t1,f1,'method',phi)：phi 用来指定 t=0 时的相位，单位为度。其余同上。

【功能介绍】 生成扫频信号。

【实例 10.7】 生成扫频信号并显示。

```
>>t=0:0.001:4;              %以 1kHz 采样频率观察 4 秒钟
>>y=chirp(t,0,2,100);       %起始时刻为直流，按 100Hz/s 线形增长
>>specgram(y,300,1e2,256,200); %显示不同时刻的功率谱
```

按照上面命令得到的扫频信号频谱如图 10.7 所示。

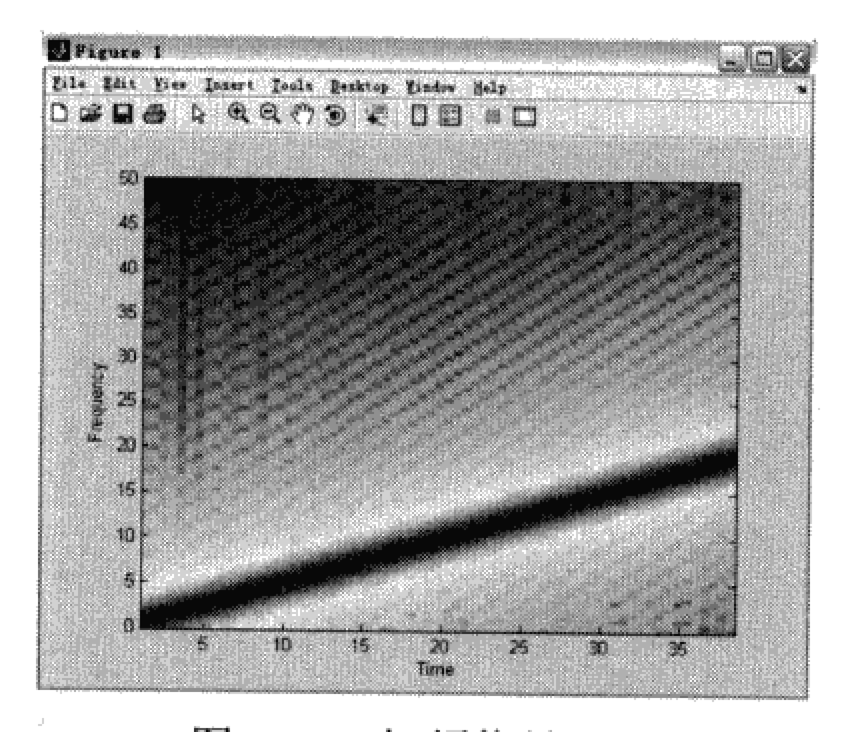


图 10.7　扫频信号显示

【实例讲解】 用户可以通过修改上面的参数数值，查看扫频信号的结果。

10.1.8　产生离散信号

【语法说明】 stem(x,y)：x、y 分别为离散点的横坐标和纵坐标。

【功能介绍】 生成离散信号。

【实例 10.8】 产生离散信号。

```
>> x=linspace(0,2*pi,20);
>> y=diric(x,6);
>> stem(x,y)
```

生成的离散点信号如图 10.8 所示。

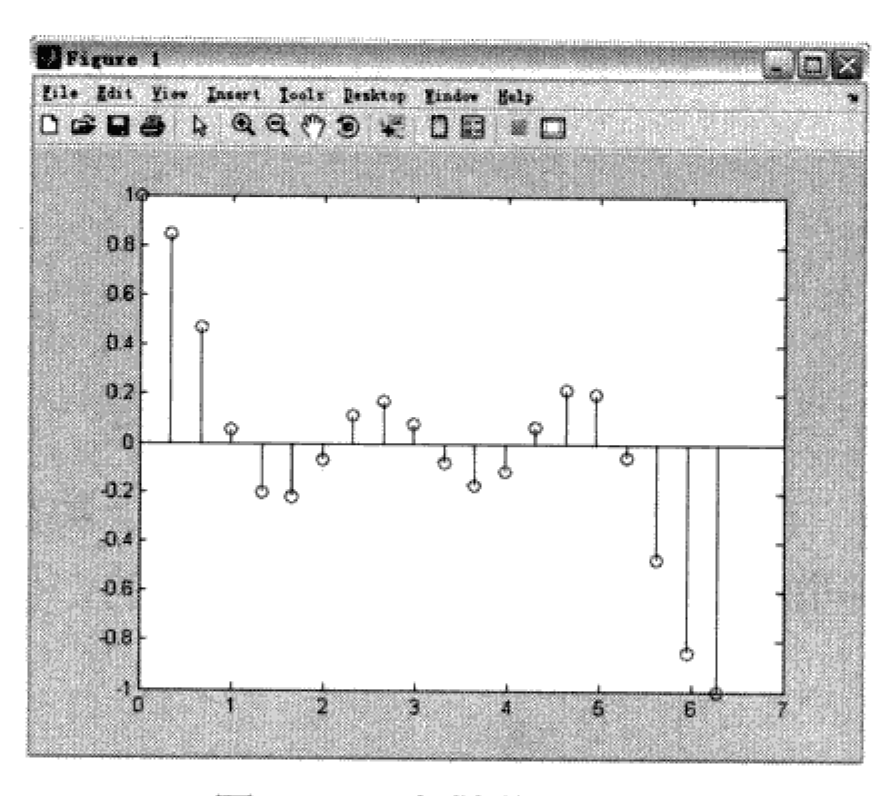


图 10.8　离散信号显示

【实例讲解】 在 MATLAB 的基础函数中，也有 stem 函数，用户可以类比。

10.2　信号的时频分析

在信号处理工具箱中，信号的产生是基础。而信号的分析处理是很重要的部分，对于一个信号的分析，通常会将其转化到不同的

域上进行，例如时域分析、频域分析以及 S 域分析等。在这个章节中将要介绍时频分析的相关内容。

10.2.1 mean 函数——求取信号的均值

【语法说明】 mean(x)：x 为信号向量。

【功能介绍】 求取均值。

【实例 10.9】 生成一组随机信号，然后计算其均值。

```
>> x1=randn(1,200);
>> x2=rand(1,200);
>> mean(x1)

ans =

  -0.0440

>> mean(x2)

ans =

   0.4969
```

【实例讲解】 在这个例子中可以看出，由于随机信号是随机产生的，所以在不同的信号可得到的数值是不同的，因此得到的均值也不尽相同。

10.2.2 std 函数——求信号的标准差

【语法说明】

- std(x) ：x 为信号向量序列。
- std(x,1)：求取信号的二阶中心矩。
- std(x,flag,dim)：当 flag=1 时，求矩阵 x 的第 dim 维的标准偏差；当 flag=0 时，求矩阵 x 的中心矩。

【功能介绍】 返回向量的标准差。如果信号 x 是服从标准正态分布的随机信号，则 std 函数的平方是其方差的标准无偏估计。

【实例 10.10】 对 1～256 之间产生的随机向量求标准差。

```
>> x=randn(1,256);
>> std(x)

ans=

   0.9587

>> std(x,1)

ans =

   0.9569

>> std(x,0)

ans =

   0.9587
>>  std(x,1,2)

ans =

   0.9569
```

【实例讲解】 上例中的最后一条命令中 dim=2，表示针对的是每个行向量。

10.2.3 xcorr 函数——估计相关性

【语法说明】

- C=xcorr(x,y)：求 x、y 向量之间的相关性。
- C=xcorr(x)：返回 x 向量自身的相关性。
- C=xcorr(x,y,'option')：option 选项用于确定无偏估计还是有偏估计。
- C=xcorr(x,y,maxlags)：maxlags 用来指定计算结果中最大延迟的值。

【功能介绍】 估计向量的相关性。

【实例 10.11】 求随机向量 x 和 y 的相关性。

```
>> x=rand(6,1);
>> y=randn(6,1);
>> c=xcorr(x,y)
c =
    0.3821
    0.5622
    0.5485
    0.3946
    0.3863
    0.9801
   -1.1918
    1.7671
   -0.8188
   -0.7793
    1.2308
```

【实例讲解】 最后得到的是 x、y 的相关系数。

10.2.4 conv 函数——卷积运算

【语法说明】 C=conv（a,b）：a、b 是要进行卷积运算的两个向量，如果长度分别为 N 和 M，并且 N>M，则计算结果的长度为 M+N−1。

【功能介绍】 求两个向量的卷积运算。

【实例 10.12】 求一个随机向量与一个已知向量的卷积。

```
>> x=randn(1,8)

x =

   -0.4326 -1.6656 0.1253 0.2877 -1.1465 1.1909 1.1892 -0.0376

>> a=[-2 3 -6]

a =

    -2     3    -6
```

```
>> conv(x,a)

ans =

  Columns 1 through 8

   0.8651 2.0335 -2.6520 9.7942 2.4040 -7.5473 8.0732 -3.5027

  Columns 9 through 10

  -7.2479    0.2258
```

【实例讲解】 随机向量得到了 8 个数值，而已知的向量 a 含有 3 个参数。最后得到的卷积结果含有 8+3−1=10 个数值。

10.2.5 cov 函数——求方差和协方差

【语法说明】

- c=cov(x)：如果 x 是一个向量，则 c 为一个标量；如果 x 是一个矩阵，那么每行表示一次观测，每列对应观测中的一个参数，c 为这些参数的协方差矩阵。
- c=cov(x,y)：求向量 x 与 y 的协方差。

【功能介绍】 求向量或矩阵的方差和协方差。

【实例 10.13】 分别求向量 x=[1 2 3 4 5 7 9 3]和矩阵 $x2=\begin{bmatrix}1 & 2 & 3\\ 3 & 7 & 5\\ -3 & 9 & -4\end{bmatrix}$ 的方差。

```
>> x=[1,2,3,4,7,9,3]

x =

     1     2     3     4     7     9     3

>> cov(x)
```

```
ans =

    8.1429

>> x2=[1 2 3;3 7 5;-3 9 -4]

x2 =

     1     2     3
     3     7     5
    -3     9    -4

>> cov(x2)

ans =

    9.3333   -5.0000   14.3333
   -5.0000   13.0000   -9.5000
   14.3333   -9.5000   22.3333
```

【实例讲解】 上面的函数和统计函数中的cov类似。

【实例10.14】 求向量x=[1 2 3 4 5 6]和向量y=[3 7 5 9 1 9]的协方差。

```
>> x=[1 2 3 4 5 6]

x =

     1     2     3     4     5     6

>> y=[3 7 5 9 1 9]

y =

     3     7     5     9     1     9
>> cov(x,y)
ans =

    3.5000    1.6000
    1.6000   10.6667
```

【实例讲解】 在信号工具箱中，向量可以理解为不同的信号源。

10.2.6 fft 函数——快速傅立叶变换

【语法说明】

- Y=fft(x)：按照基 2 的时间抽取快速算法计算信号 x 的傅立叶变换，当 x 的长度为 2 的整数次幂或者 x 全为实数时，计算的时间会因此缩短。
- Y=fft(x,n)：补零的傅立叶变换。将 x 的尾部补零使 x 的长度达到 n，然后对补零后的数据进行快速傅立叶变换。

【功能介绍】 快速傅立叶变换。

【实例 10.15】 对于含有正余弦信号并且叠加噪声的信号进行傅立叶变换，并求得原始信号的频率成分。

```
>> t=0:0.001:0.8;
>> x=cos(2*pi*30*t)+sin(2*pi*100*t);
>> y=x+randn(1,length(t));            %含有噪声的信号数据
>> plot(y(1:40))                      %绘制原始信号图
>> Y=fft(y,512);                      %傅立叶变换求解
>> P=Y.*conj(Y)/512;                  %求取功率密度
>> f=1000*(0:255)/512;                %设定频率变化范围
>> plot(f,P(1:256))                   %绘制频谱图
```

得到的结果如图 10.9 和图 10.10 所示。

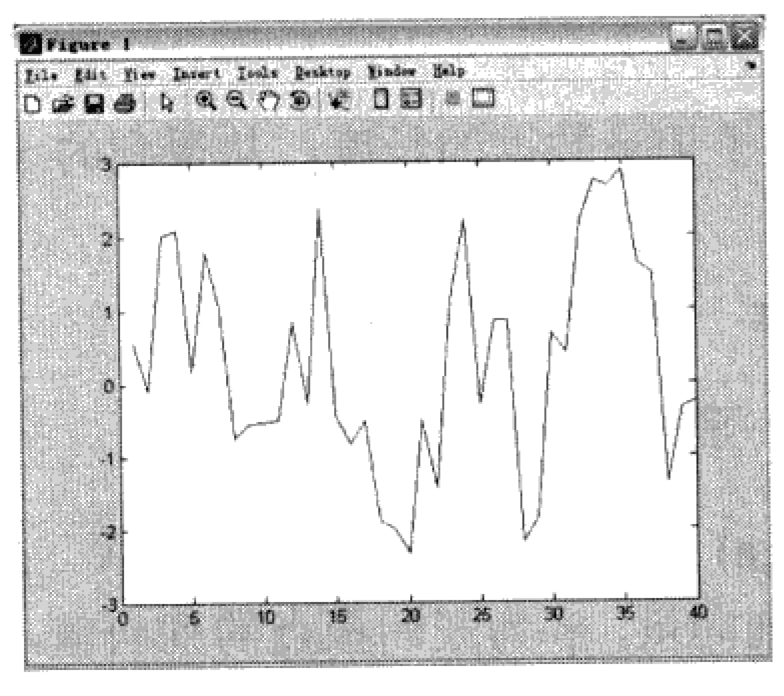


图 10.9　原始信号

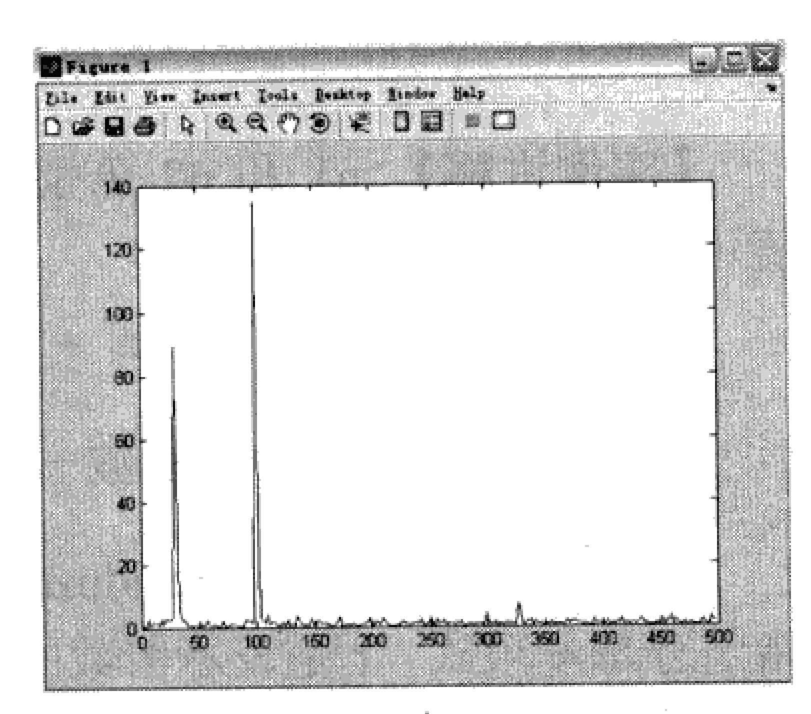


图 10.10　功率谱密度分布

【实例讲解】 在第一幅图中很难找到信号的原形，而通过第二个功率谱密度图则容易看出，信号中含有频率为 30Hz 和 100Hz 的成分。

10.2.7 离散信号的 Z 变换

【语法说明】 利用卷积求取 Z 变换。

【实例 10.16】 设 $X1(z)=z-3+4z^{-1}, X2(z)=3z^2-4z+9-4z^{-1}$，求 $X(z)=X1(z)X2(z)$。

```
>> x1=[1 -3 4];
>> n1=-1:1;
>> x2=[3 -4 9 -4];
>> n2=-3:1;
>> x=conv(x1,x2)

x =


     3   -13    33   -47    48   -16
>> ns=n1(1)+n2(1);
>> ne=n1(length(x1))+n2(length(x2));
>> n=ns:ne;
>> n=ns:ne

n =


    -4    -3    -2    -1     0     1
```

【实例讲解】 n1 和 n2 分别用于记录数据的长度，从得到的 x 值可以知道，最终结果为：

$$X(z)=3z^4-13z^3+33z^2-47z+48-16z^{-1}$$

10.2.8 residuze 函数——离散信号的 Z 反变换

【语法说明】 [r,p,k]=residuze(b,a)：向量 b 指定分子多项式，a 指定分母多项式，r 是表示留数的列向量，p 为表明极点的列向量，k 表明展开式中的直接项。

【功能介绍】 求 Z 反变换。

【实例 10.17】 对于 $X(z)=\dfrac{3-\frac{5}{6}z^{-1}}{1-\frac{7}{12}z^{-1}+\frac{1}{12}z^{-2}},\frac{1}{4}<|z|<\frac{1}{3}$，求 Z 的反变换。

```
>> [r,p,k]=residuez([3,-5/6],[1,-7/12,1/12])

r =

    2.0000
    1.0000


p =

    0.3333
    0.2500


k =

     []
```

【实例讲解】 因此，结果得到的反变换为：

$$x(n)=(\frac{1}{4})^n u(n)-2(\frac{1}{3})^n u(-n-1)$$

10.2.9 hilbert 函数——希尔伯特变换

【语法说明】 y=Hilbert(x)。

【功能介绍】 此函数用来计算实序列的希尔伯特变换并且返回一个同样长度的复数序列。

【实例 10.18】 对频率为 50Hz 的正弦函数进行希尔伯特变换。

```
>> t=0:1/511:1;
>> x=sin(2*pi*50*t);
```

```
>>  y=hilbert(x);
>>  plot(t(1:40),real(y(1:40))),hold on
>> plot(t(1:40),imag(y(1:40)))
```

上例中得到的图形如图 10.11 所示。

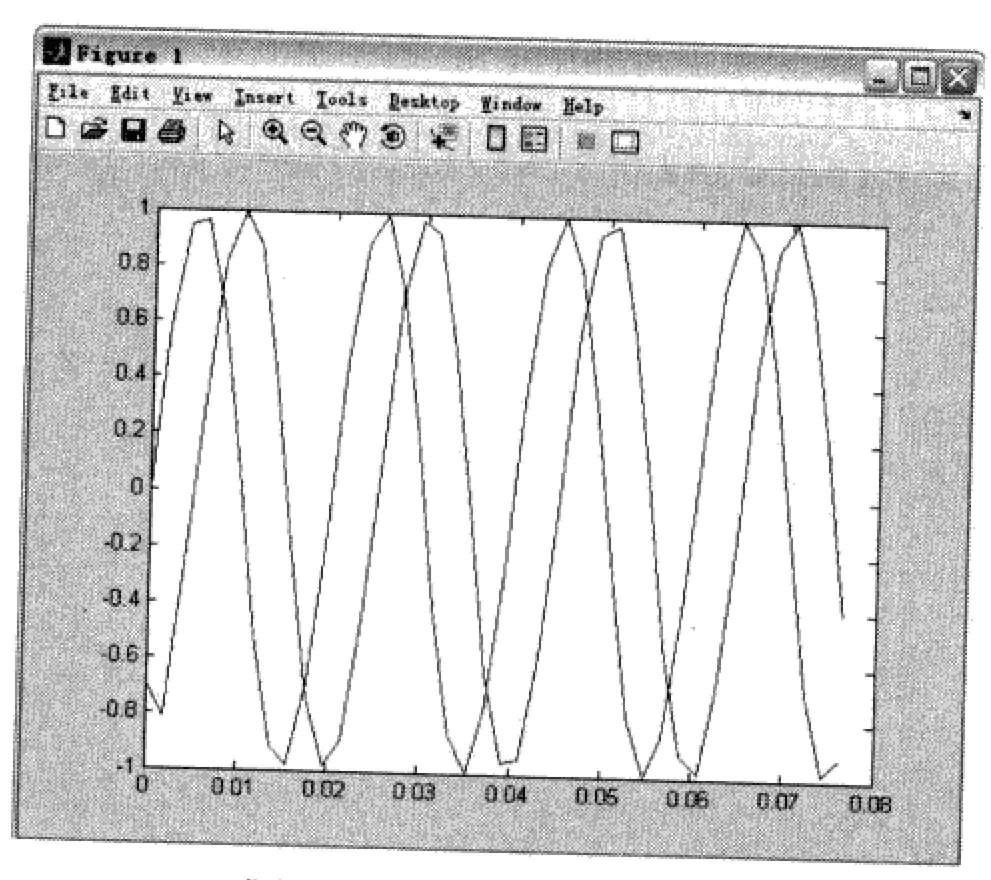


图 10.11　希尔伯特变换

【实例讲解】 返回的图形中，y 的实部为原始信号，y 的虚部为希尔伯特变换序列。

10.3　滤波器的设计

数字滤波器的设计是信号处理技术的一个重要问题，在这一节中将主要介绍在 MATLAB 中如何实现数字滤波器的设计，并且利用信号处理的有关理论知识以及前面章节介绍的 MATLAB 函数来设计各种数字滤波器。

10.3.1　buttap 函数——设计巴特沃思滤波器

【背景知识】 butterworth 滤波器的特点是具有通带内最大平滑

的幅度特性，而且随着频率升高呈单调减小。通常，butterworth 滤波器又称“最平”的幅频响应滤波器。

【语法说明】 [z,p,k]=buttap(n)：返回一个 n 阶 butterworth 滤波器的零点、极点和增益。

【功能介绍】 设计巴特沃思滤波器，并给出 butterworth 滤波器的零点、极点和增益。

【实例 10.19】 设计一个 20 阶的巴特沃思滤波器。

```
>> [z,p,k]=buttap(20)

z =

     []
p =

  -0.0785 + 0.9969i
  -0.0785 - 0.9969i
  -0.2334 + 0.9724i
  -0.2334 - 0.9724i
  -0.3827 + 0.9239i
  -0.3827 - 0.9239i
  -0.5225 + 0.8526i
  -0.5225 - 0.8526i
  -0.6494 + 0.7604i
  -0.6494 - 0.7604i
  -0.7604 + 0.6494i
  -0.7604 - 0.6494i
  -0.8526 + 0.5225i
  -0.8526 - 0.5225i
  -0.9239 + 0.3827i
  -0.9239 - 0.3827i
  -0.9724 + 0.2334i
  -0.9724 - 0.2334i
  -0.9969 + 0.0785i
  -0.9969 - 0.0785i
```

```
k =

     1
```

【实例讲解】 设计得到的这个滤波器没有零点，极点为上面显示的复数序列，增益为 1，表明了这个滤波器没有对信号进行放大和缩小的操作。

10.3.2 cheb1ap 函数——设计 Chebyshev 1 低通模拟滤波器

【语法说明】 [z,p,k]=cheb1ap(n,rp)：返回一个 n 阶 Chebyshev 1 型滤波器的零点、极点和增益。这个滤波器在通带内的最大衰减为 rp。

【功能介绍】 设计 Chebyshev 1 型低通模拟滤波器。

【实例 10.20】 生成一个 6 阶 Chebyshev 1 型低通模拟滤波器，使其在通带内的最大衰减为 0.2dB。

```
>> n=6

n =

     6

>> rp=0.2

rp =

    0.2000
>> [z,p,k]=cheb1ap(n,rp)

z =

    []


p =
```

```
  -0.0985 + 1.0335i
  -0.2692 + 0.7566i
  -0.3677 + 0.2769i
  -0.3677 - 0.2769i
  -0.2692 - 0.7566i
  -0.0985 - 1.0335i


k =

   0.1439
```

【实例讲解】 从上面的结果可以看出，这个滤波器有零点，同时最大的增益是 0.1439。

10.3.3 cheb2ap 函数——设计 Chebyshev 2 型滤波器

【语法说明】 [z,k,p]=cheb2ap(n,rs)：返回一个 n 阶 Chebyshev 2 型模拟低通滤波器。

【功能介绍】 设计 Chebyshev 2 型滤波器。

【实例 10.21】 设计一个 6 阶 Chebyshev 2 型模拟低通滤波器，其带阻内的最小衰减为 rs=60dB。

```
>> n=6

n =

     6

>> ns=60

ns =

    60

>> [z,p,k]=cheb2ap(n,ns)
```

```
z =

        0 + 1.0353i
        0 - 1.0353i
        0 + 1.4142i
        0 - 1.4142i
        0 + 3.8637i
        0 - 3.8637i
p =

  -0.1174 - 0.5136i
  -0.3645 - 0.4274i
  -0.5767 - 0.1812i
  -0.5767 + 0.1812i
  -0.3645 + 0.4274i
  -0.1174 + 0.5136i
k =
 1.0000e-003
```

【实例讲解】 得到的零点和极点为 z、p，增益为 k 所表示的值。

10.3.4 besselap 函数——设计 Bessel 低通滤波器

【语法说明】 [z,p,k]=besselap(n)：返回 n 阶带通滤波器的零点、极点和增益值。

【功能介绍】 设计 Bessel 低通滤波器。

【实例 10.22】 生成一个 8 阶低通模拟滤波器。

```
>> [z,p,k]=besselap(8)

z =

     []
p =

  -0.9097 - 0.1412i
  -0.9097 + 0.1412i
  -0.8473 - 0.4259i
```

```
  -0.8473 + 0.4259i
  -0.7111 - 0.7187i
  -0.7111 + 0.7187i
  -0.4622 - 1.0344i
  -0.4622 + 1.0344i
k =

    1
```

【实例讲解】 这个8阶低通滤波器的零点、极点和增益分别由z、p、k表示。

10.3.5 butter函数——设计Butterworth滤波器

【语法说明】

- [B,A]=butter(n,Wn)：设计一个n阶的低通Butterworth[]滤波器，并且返回滤波器系数矩阵[B，A]。其中固有频率Wn必须是在[0,1]范围之内的归一化频率，它的最大值为采样频率的1/2，MATLAB中默认为2Hz。
- [B，A]=butter(n,Wn,'high')：设计一个n阶高通滤波器。
- [B，A]=butter(n,Wn,'stop')：设计一个n阶带阻滤波器。

【功能介绍】 此函数不仅可以设计模拟滤波器，还可以设计数字滤波器。

【实例10.23】 设计一个模拟Butterworth滤波器，截至频率为fp=4000Hz，通带最大衰减为ap=3dB，阻带起始频率fs=10000Hz，阻带最小衰减as=30dB。

```
>> wp=5000*2*pi;                    %通带结束频率
>> ws=1000*2*pi;                    %阻带开始频率
>> ap=3;                            %通带最大衰减
>> as=-30;                          %阻带最小衰减
>> [N,Wn]=buttord(wp,ws,ap,as,'s'); %获得低通Butter worth
滤波器阶数和通带截至频率
>> [b,a]=butter(N,Wn,'s');
>> freqs(b,a);                      %计算频率响应
```

得到的图形如图10.12所示。

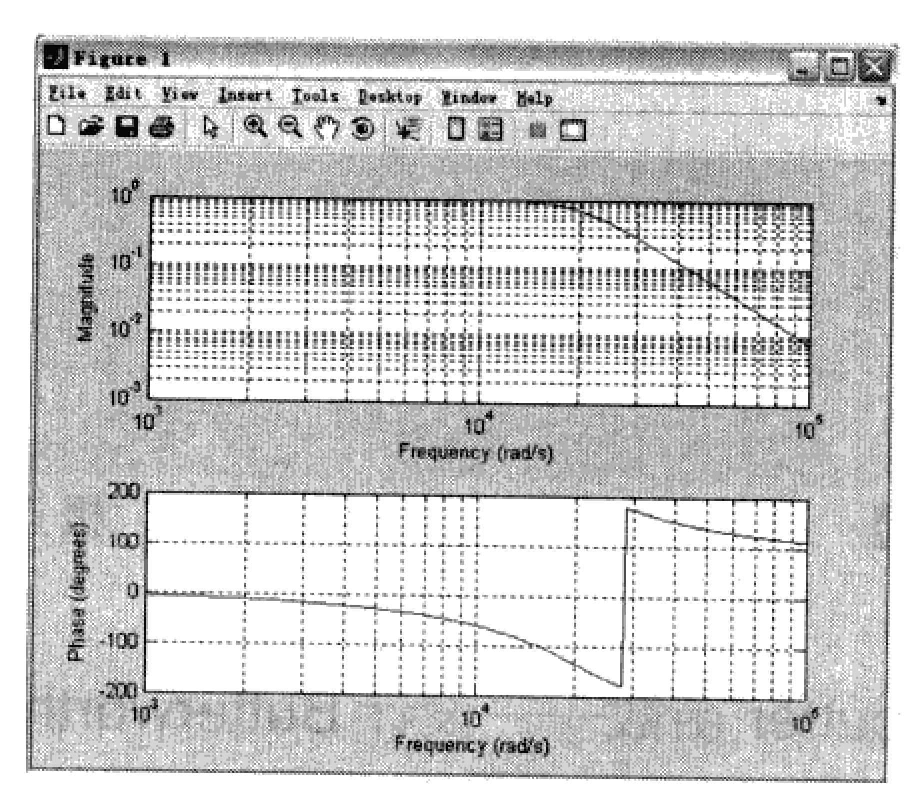


图 10.12　设计 Butterworth 滤波器

【实例讲解】 用户可以直接从上面的滤波器中查看出波形的特征值。

【实例 10.24】 设计一个模拟低通滤波器 Butterworth 滤波器，要求：通带截至频率 f_p=2000Hz，带内最大衰减 R_p=5dB，阻带截至频率 f_s=4000Hz，带内最小衰减 R_s=40dB。

```
>> clear
>> Wp=2000*2*pi;                          %通带结束频率
>> Ws=400082*pi;                          %阻带开始频率
>> Rp=5;                                  %通带最大衰减
>> Rs=40;                                 %阻带最小衰减
>> [n,Wn]=buttord(Wp,Ws,Rp,Rs,'s');      %获得低通 Butterworth
滤波器阶数和通带截至频率
>> [z,p,k]=buttap(n);                     % 设 计 模 拟 低 通
Butterworth 滤波器
>> [b0,a0]=zp2tf(z,p,k);                  %转化为传递函数表示滤波器
>> [b,a]=lp2lp(b0,a0,Wn);                 %将滤波器原型变换为要求设
计的低通滤波器
>> [h,w]=freqs(b,a);                      %计算频率响应
>> plot(w/(2*pi),20*log10(abs(h)));
>> grid on;
>> xlabel('频率(Hz)');
>> ylabel('幅频响应(dB)');
```

得到的图形如图 10.13 所示。

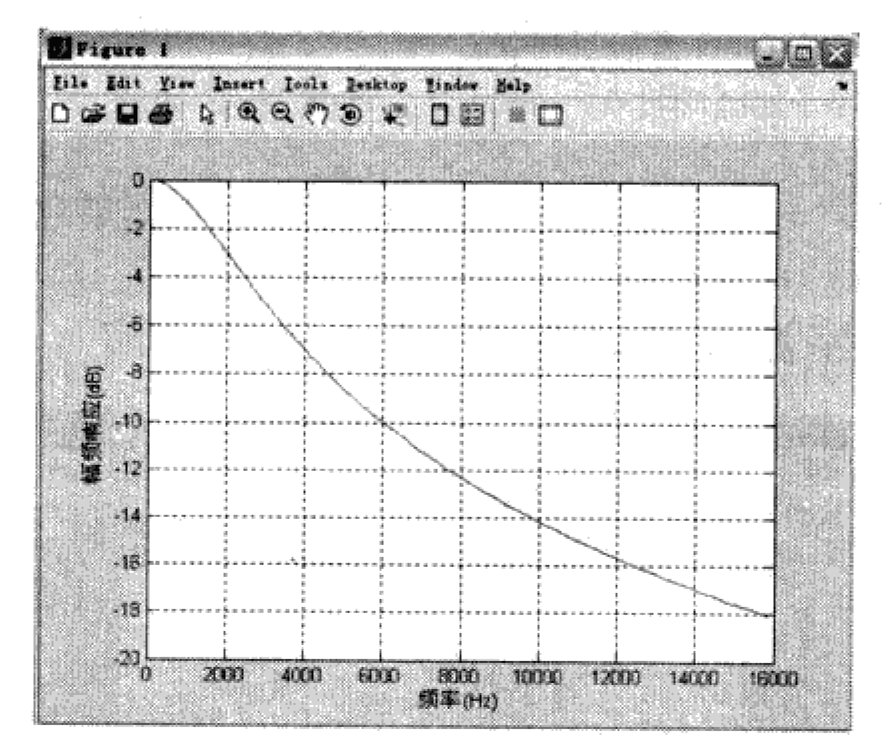


图 10.13　n 阶模拟低通 Butterworth 滤波器

【实例讲解】 从上面的图形中，用户可以直接查看到波形的特点。

10.3.6　impinvar 函数——模拟滤波器转化为数字滤波器

【语法说明】

- [bz,az]=impinvar(b,a,fs)：参数 a、b 给出模拟滤波器传递函数的分子系数和分母系数；f_s 是采样频率，缺省情况下是 1Hz；bz 和 az 返回设计出的数字滤波器的传递函数的分子和分母系数。
- [bz,az]=impinvar(b,a) ：同上。

【功能介绍】 将模拟滤波器转化为数字滤波器。

【实例 10.25】 将如下模拟滤波器转化为数字滤波器，采样频率为 20kHz。低通滤波器的要求如下：

通带截至频率 f_p=2000Hz，带内最大衰减 R_p=5dB；

阻带截至频率：f_s=4000Hz，带内最小衰减 R_S=40dB。

```
>> clear
>> wp=2000*2*pi;
>> ws=4000*2*pi;
>> rp=5;
>> rs=40;
>> Fs=20000;                          %采样频率
>> [n,Wn]=buttord(wp,ws,rp,rs,'s');
>> [z,p,k]=buttap(n);
```

```
>> [b0,a0]=zp2tf(z,p,k);      %转化为传递函数表示滤波器
>> [b,a]=lp2lp(b0,a0,Wn);     %将滤波器原型变换为要求设计的低通滤波器
>> [bz,az]=impinvar(b,a,Fs); %模拟滤波器转化为数字滤波器
>> freqz(bz,az);
```

转化后的数字滤波器如图 10.14 所示。

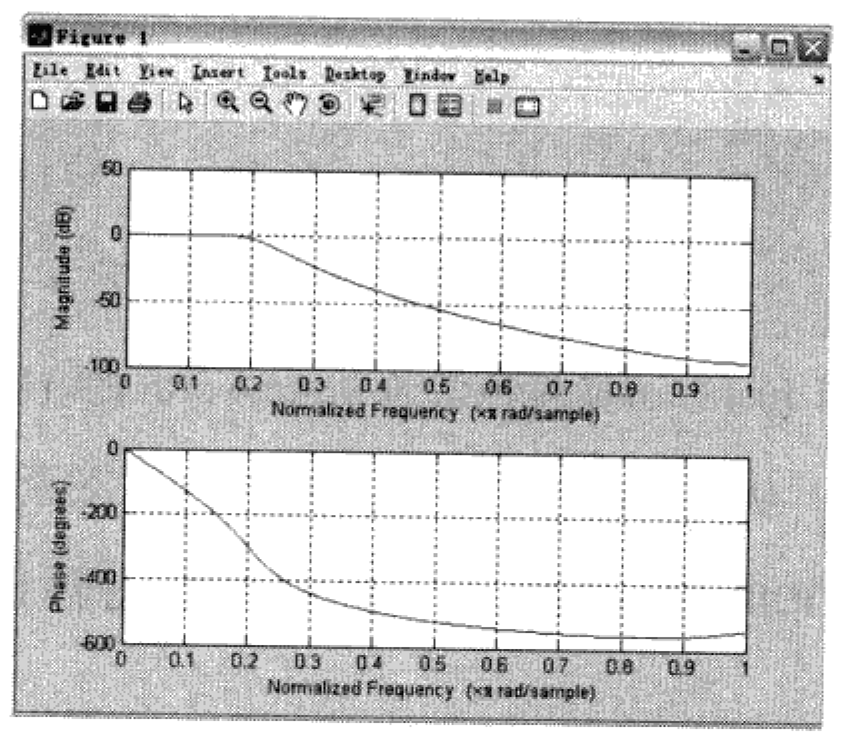


图 10.14　模拟滤波器转化为数字滤波器

【实例讲解】 从上面的结果中可以对比模拟和数字滤波器的转换过程。

10.3.7　bilinear 函数——用双线形变换法将模拟滤波器转化为数字滤波器

【语法说明】

- [zd,pd,kd]=bilinear(z,p,k,fs)：z、p、k 分别为模拟滤波器的零点、极值和增益，fs 为频率。
- [numd,dend]=bilinear(num,den,fs)：num 和 den 分别表示滤波器传递函数的分子与分母多项式系数向量，fs 为频率。

【功能介绍】 用双线形变换法将模拟滤波器转化为数字滤波器。

【实例 10.26】 将上例中的模拟滤波器用双线形变换法转化为

数字滤波器，采样频率为20kHz。

```
>> clear
>> wp=2000*2*pi;
>> ws=4000*2*pi;
>> rp=5;
>> rs=40;
>> Fs=20000;
>> [n,Wn]=buttord(wp,ws,rp,rs,'s');
>> [z,p,k]=buttap(n);
>> [b0,a0]=zp2tf(z,p,k);
>> [b,a]=lp2lp(b0,a0,Wn);
>> [bz,az]=bilinear(b,a,Fs);
>> freqz(bz,az);
```

转化后的数字滤波器如图10.15所示。

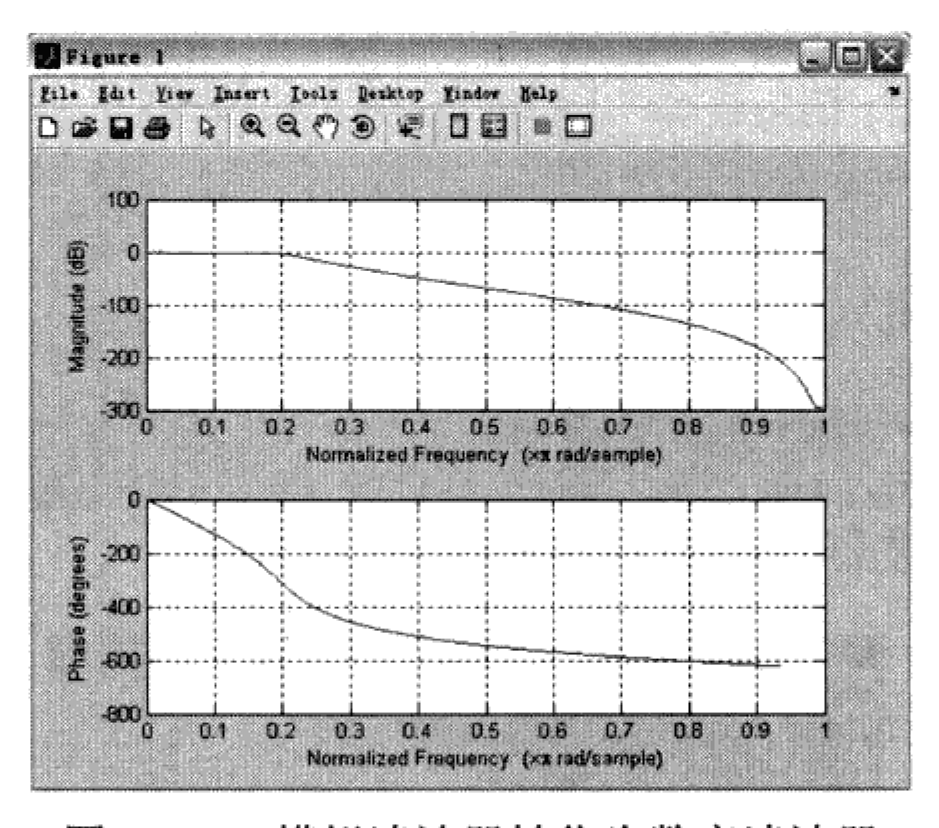


图10.15 模拟滤波器转化为数字滤波器

【实例讲解】 这个函数和上面一个函数的差别是转换的方法不同。

10.3.8 cheby1函数——设计Chebyschev 1型滤波器

【语法说明】

- [b,a]=cheby1(n,Rp,Wn,options)：设计一个n阶的低通Butterworth滤波器，并且返回滤波器系数矩阵[a，b]。其中固有频

率 Wn 必须是在[0，1]范围之内的归一化频率，它的最大值为采样频率的 1/2，MATLAB 中默认为 2Hz。Rp 用来指定通带允许的纹波。

- [z,p,k]=cheby1(n,Rp,Wn,options) ：用法同上。
- [A,B,C,D]=cheby1(n,Rp,Wn,options)：用法同上。

【功能介绍】 设计 Chebyschev1 型数字滤波器。

【实例 10.27】 设计一个数字低通滤波器，要求在通带 0～0.2π 内纹波不大于 3dB，在阻带 0.6π～π内衰减不小于 20dB。

```
>> [n,Wn]=cheb1ord(0.2,0.6,3,20)  %利用这个函数取得给定频率特性下的滤波器阶数和通带截至频率

n =

     2


Wn =

    0.2000

>> [b,a]=cheby1(n,3,Wn);
>> freqz(b,a);
```

设计的滤波器的频率特性如图 10.16 所示。

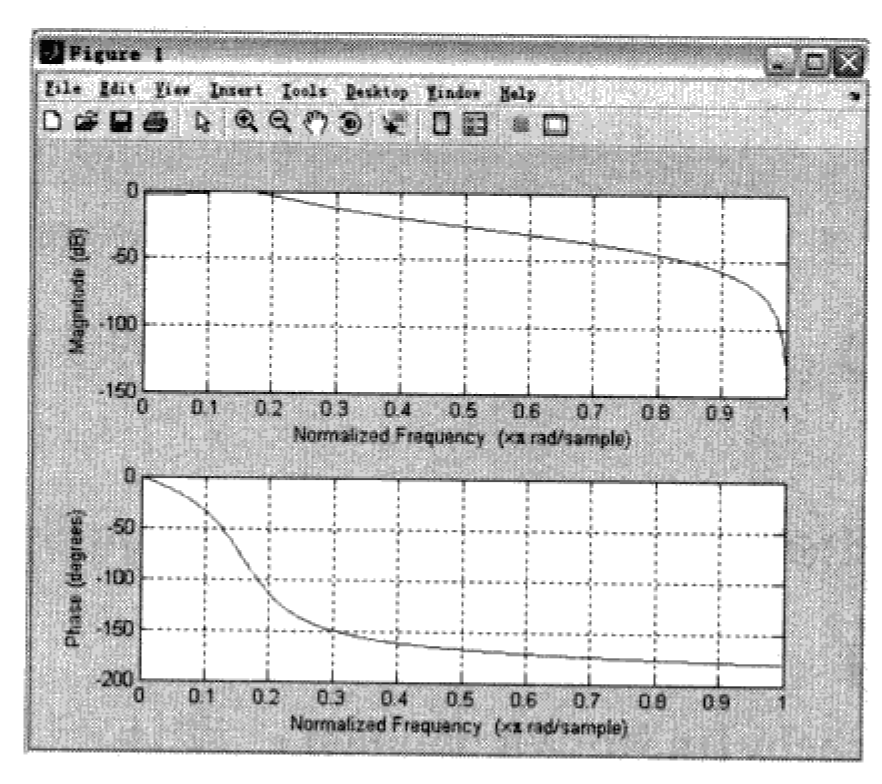


图 10.16　Chebyschev 1 型低通滤波器的设计

【实例讲解】 用户可以修改滤波器的参数，对比设计的结果。

10.3.9 cheby2 函数——设计 Chebyschev 2 型滤波器

【语法说明】

- [b,a]=cheby2(n,Rs,Wn)。
- [b,a]=cheby2(n,Rs,Wn,'ftype')。
- [b,a]=cheby2(n,Rs,Wn, 's')。
- [b,a]=cheby2(n,Rs,Wn, 'ftype', 's')。
- [z,p,k]=cheby2(n,Rp,Wn,options)。
- [A,B,C]=cheby2(n,Rp,Wn,options)。

用法与 cheby1 相同，不再赘述。

【功能介绍】 设计 Chebyschev 2 型滤波器。

【实例 10.28】 设计一个 Chebyschev 2 型低通数字滤波器，采样频率为 1000Hz，要求：

通带截至频率为 30Hz，带内最大衰减为 5dB；阻带截至频率为 120Hz，带内最小衰减为 50dB。

```
>> clear;
>> Fs=1000;                      %设置采样频率
>> Wp=2*Fs*tan(30*pi/Fs);        %通带截至频率
>> Ws=2*Fs*tan(120*pi/Fs);       %阻带截至频率
>> Rp=5;                         %带内最大衰减
>> Rs=50;                        %带内最小衰减
>> [n,Wn]=cheb2ord(Wp,Ws,Rp,Rs,'s');  %取得给定频率下的滤波器阶数和通带截至频率
>> [z,p,k]=cheb2ap(n,Rs);        %设计滤波器原型
>> [b0,a0]=zp2tf(z,p,k);         %用传递函数表示滤波器
>> [b,a]=lp2lp(b0,a0,Wn);        %将滤波器原型变换为要求设计的低通滤波器
>> [b1,a1]=bilinear(b,a,Fs);
>> freqz(b1,a1);
```

设计完成的 Chebyschev 2 型低通滤波器设计如图 10.17 所示。

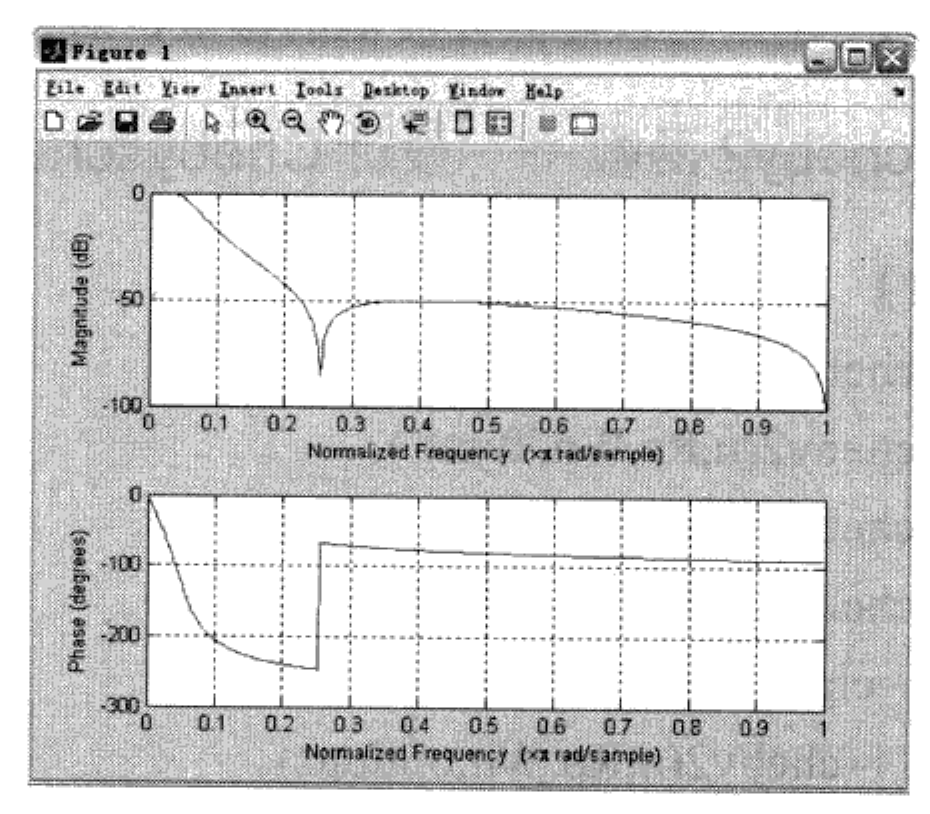


图 10.17　Chebyschev2 型低通滤波器设计

【实例讲解】 用户可以将这个函数和上面的函数进行对比。

10.3.10　ellip 函数——设计椭圆形滤波器

【语法说明】

- [b,a]=ellip(n,Rp,Rs,Wn)。
- [b,a]=ellip(n,Rp,Rs,Wn,'ftype')。
- [b,a]=ellip(n,Rp,Rs,Wn, 's')。
- [b,a]=ellip(n,Rp,Rs,Wn, 'ftype', 's')。
- [z,p,k]=ellip (n,Rp,Wn,options)。
- [A,B,C]= ellip (n,Rp,Wn,options)。

用法与 cheby1 相同，不再赘述。

【功能介绍】 设计椭圆形滤波器。

10.3.11　bessel 函数——设计 bessel 滤波器

【语法说明】

- [b,a]=bessel(n,Wn)。
- [b,a]=ellip(n,Wn, 'ftype')。

- [z,p,k]=ellip (n,Rp,Wn,options)。
- [A,B,C]= ellip (n,Rp,Wn,options)。

【功能介绍】 设计 bessel 滤波器。

【实例 10.29】 设计一个 10 阶的低通滤波器，用来滤除频率高于 6000 的信号频率成分。

```
>> [b,a]=besself(10,6000);
>> freqz(b,a);
```

得到的频率响应曲线如图 10.18 所示。

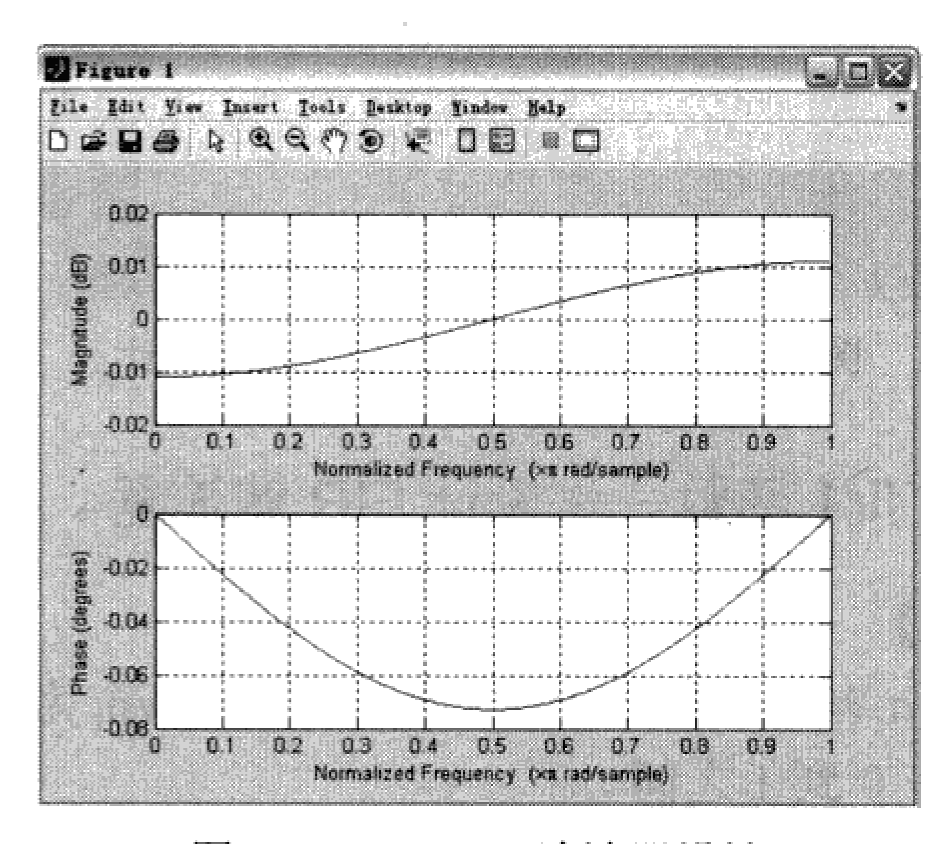


图 10.18　Bessel 滤波器设计

10.3.12　yulewalk 函数——设计 yulewalkIIR 型滤波器

【语法说明】 [b,a]=yulewalk(n,f,m) ：[f,m]为系统的幅频响应，f 为一数组，其元素从 0 到 1 递增。

【功能介绍】 设计 IIR 型滤波器。

【实例 10.30】 设计一个 12 阶的 yulewalk 滤波器。

```
>> f=[0 0.6 0.6 1];
>> m=[1 1 0 0];
>> [b,a]=yulewalk(10,f,m);
>> [h,w]=freqz(b,a,512);
>>  plot(f,m,'-',w/pi,abs(h),'-');
```

设计完成的滤波器如图 10.19 所示。

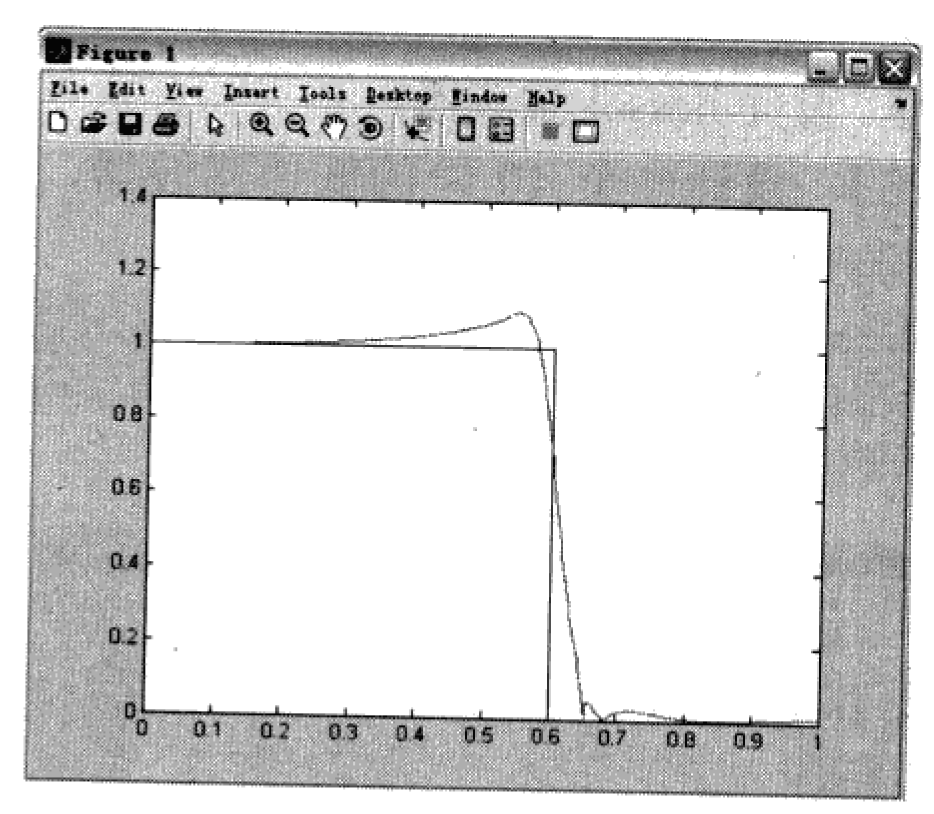


图 10.19　yulewalk IIR 型滤波器设计

10.3.13　fir1 函数——设计 FIR 滤波器

【语法说明】

- b=fir1(n,Wn)：n 指定滤波器的阶数，Wn 为滤波器的截至频率，其取值在 0 到 1 之间。
- b=fir1(n,Wn, 'ftype')。
- b=fir1(n,Wn,window)：window 参数指定使用的窗向量，例如可以取值为 Boxcar、Hanning、Bartlett、Blackman、Kaiser 及 Chebwin。
- b=fir1(n,Wn, 'ftype',window)：ftype 为滤波器的类型，其取值可为如下值。
- ftype='high'：指定一个具有截至频率 Wn 的高频滤波器。
- Ftype='stop'：指定一个带阻滤波器，其阻带频率为 Wn=[w1,w2]。
- Ftype='DC-0'：在多频带滤波中，使第一个频带 0<w<w1 为阻带。

- Ftype='DC-1'：在多频带滤波中，使第一个频带 0<w<w1 为通带。

【功能介绍】 设计 FIR 滤波器。

【实例 10.31】 设计一个低通滤波器，采样频率为 6000Hz，满足如下的要求：通带范围 0Hz～1000Hz ，带内波动 3%，阻带范围 1200Hz～3600Hz，带内最小衰减为 Rs=30dB。

```
>> clear;
>> Fs=6000;
>> fzu0=[1000 1200];
>> g=[1 0];
>> bod=[0.05 0.1]

bod =

    0.0500    0.1000

>> [n,Wn,beta,ftype]=kaiserord(fzu0,g,bod,Fs); %利用Kaiser 窗设计 FIR 滤波器,首先进行有关参数计算
>> h=fir1(n,Wn,ftype,kaiser(n+1,beta));
>> freqz(h);
```

得到的 FIR 低通滤波器如图 10.20 所示。

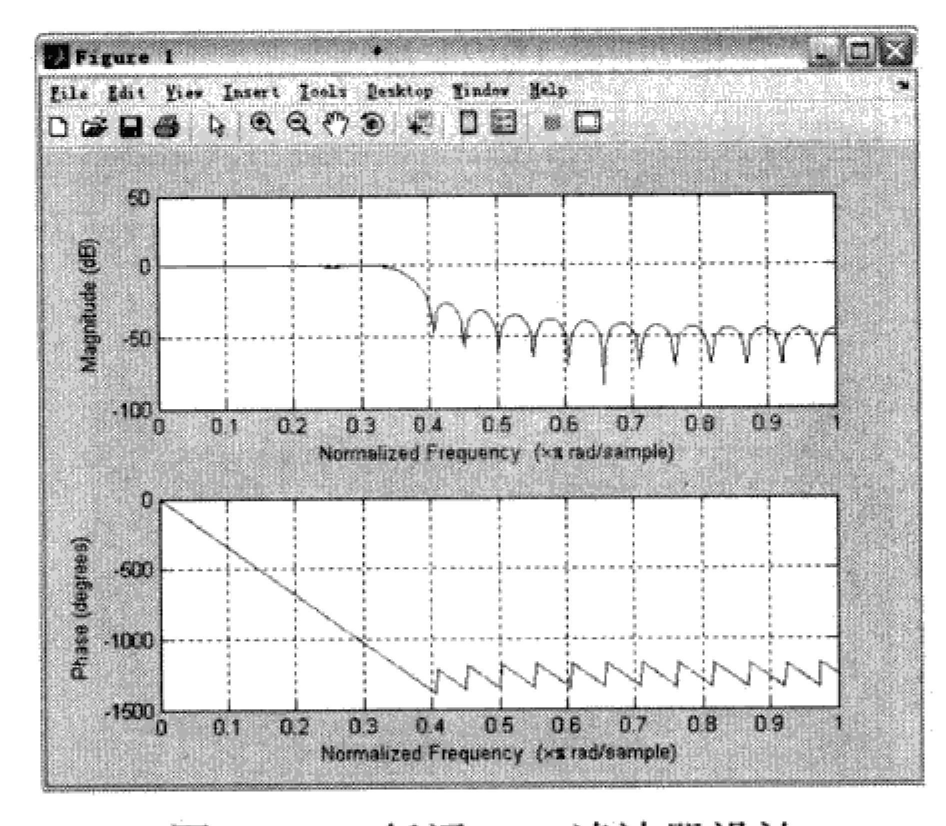


图 10.20 低通 FIR 滤波器设计

10.3.14 fir2 函数——利用窗口法进行 FIR 滤波器设计

【语法说明】

- B=fir2(n,f,m)。
- B=fir2(n,f,m,window)。
- B=fir2(n,f,m,npt)。
- B=fir2(n,f,m,npt,window)。
- B=fir2(n,f,m,npt,lap)。
- B=fir2(n,f,m,npt,lap,window)。

f 给出一组升序排列的频率值，m 给出对应于每个频率值的幅频响应值。npt 参数给出了总的频率取值点数，默认为 512 点，lap 是 f 中重复取值的点数，默认值为 25，lap/npa 决定了滤波器的过滤带宽。

【功能介绍】 利用窗口法进行 FIR 滤波器的设计。

【实例 10.32】 利用 Hamming 窗设计一个 36 阶的双通带滤波器，第一个通带从 0.1π～0.3π，第二个通带从 0.6π～0.9π。

```
>>clear;
>> f=[0 0.1 0.1 0.2 0.3 0.3 0.4 0.5 0.6 0.6 0.7 0.8 0.8 1];
>> m=[0 0 1 1 1 0 0 0 1 1 1 0 0 0];
>> b=fir2(36,f,m,300,30);
>> freqz(b,1,256);
```

得到的双通带滤波器如图 10.21 所示。

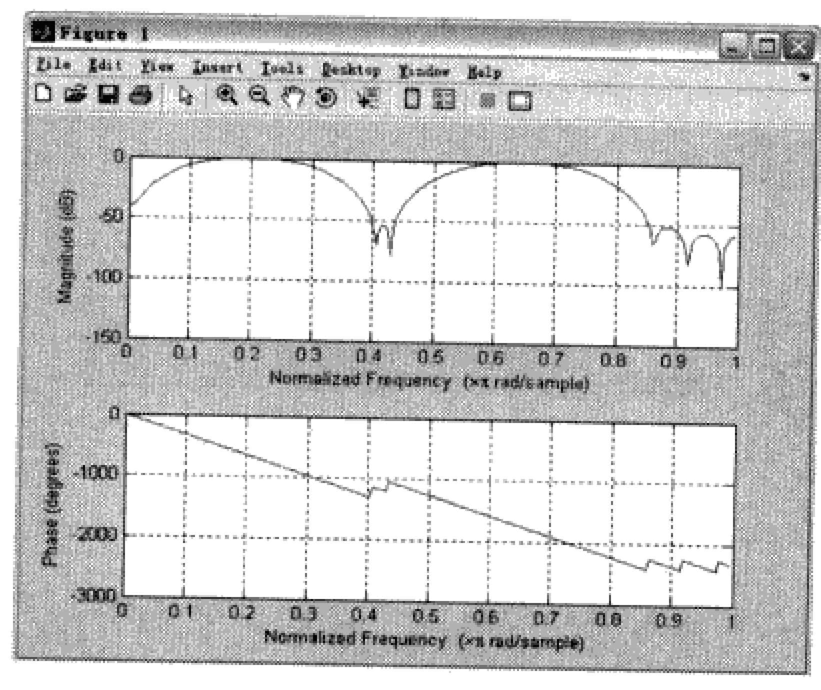
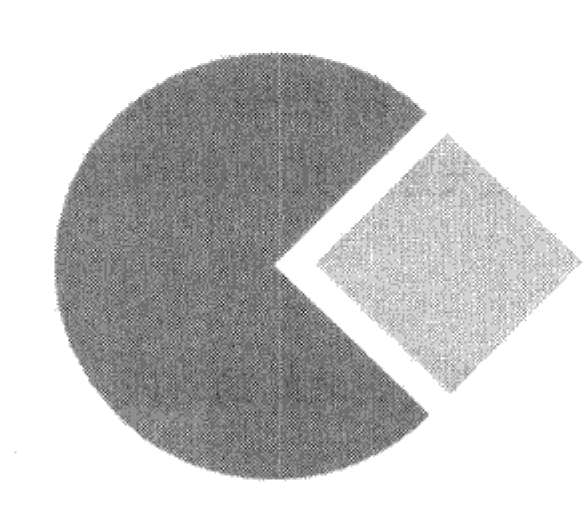


图 10.21 Hamming 窗设计的 FIR 双通带滤波器



第 11 章　符号数学工具箱

MATLAB 符号运算是通过集成在 MATLAB 中的符号数学工具箱来实现的。和别的工具箱有所不同，该工具箱不是基于矩阵的数值分析，而是使用字符串来进行符号分析与运算。MATLAB 的符号数学工具箱可以完成几乎所有符号运算功能。其中有符号表达式运算，符号表达式的复合、化简，符号矩阵的运算，符号微积分，符号函数画图，符号微分方程求解等。同时，符号数学工具箱还支持可变精度运算。

11.1　符号表达式的 MATLAB 表示

符号数学工具箱是操作和解决符号表达式的符号数学函数集合，那么符号表达式在 MATLAB 中是怎样表示的呢？这一节将要简单介绍。

符号表达式是代表数字、函数、算子和变量的 MATLAB 字符串，或字符串数组，不要求变量有预先确定的值。符号方程式是含有等号的符号表达式。符号算术是使用已知的规则和给定符号恒等式求解这些符号方程的实践，它与代数和微积分所学到的求解方法完全一样。符号矩阵是数组，其元素是符号表达式。

MATLAB 把符号表达式表示成字符串，以与数字变量或运算相区别；否则，这些符号表达式会和基本的 MATLAB 函数相混淆。表 11.1 中的这几个符号表达式代表了基本的转换关系，其余的可以参照这个表格来仿写 MATLAB 表达式。

表 11.1　　符号表达式与 MATLAB 表达式之间的对应

符号表达式	MATLAB 表达式
$M=\begin{bmatrix} a & b \\ c & d \end{bmatrix}$	M=sym(' [a，b；c，d] ')
$\frac{1}{2x^n}$	' 1/(2*x^n) '
$y=\frac{1}{\sqrt{2x}}$	y= ' 1/sqrt(2*x) '
$\cos(x^2)-\sin(2x)$	' cos(x^2)-sin(2*x) '
$\int_a^b \frac{x^3}{\sqrt{1-x}}dx$	f=int(' x^3/sqrt(1-x) ' ， ' a ' ， ' b ')

11.2 符号表达式的运算

运用符号表达式，不仅仅是为了创建一个新的字符串就可以，而是想以某些方式改变它；也许希望提取表达式的一部分，也可能希望合并两个表达式或求得表达式的数值。这样，就需要对表达式的运算做研究，在 MATLAB 中有许多符号工具可以帮助完成这些任务。所有符号函数作用到符号表达式和符号数组，都会返回符号表达式或数组。其结果有时可能看起来像一个数字，但事实上它是一个内部用字符串表示的符号表达式。

11.2.1 numden 函数——提取分子和分母

【语法说明】

- [n,d]=numden(a) ：提取符号表达式 a 的分子和分母，并将其存放在 n 和 d 中。
- n=numden(a)：提取符号表达式 a 的分子和分母，只将分子存放在 n 中。

【功能介绍】 对符号表达式提取分子和分母的操作。

【实例 11.1】 给定如下表达式，用 numden 函数对其进行合并、有理化并返回所得的分子和分母。

$$m = x^2 \quad f = \frac{ax^2}{b-x} \quad g = \frac{3}{2}x^2 + \frac{2}{3}x - \frac{3}{5}$$

$$h = \frac{x^2+3}{2x-1} + \frac{3x}{x-1} \quad k = \begin{bmatrix} \frac{3}{2} & \frac{2x+1}{3} \\ \frac{4}{x^2} & 3x+4 \end{bmatrix}$$

```
>> m= ' x^2 '   %  创建一个简单表达式
m=
 x^2

>> [n,d]=numden(m)
n=
 x^2
d=
 1

>> f= ' a*x^2/(b-x) '   %  建立一个符号表达式
f=
 a*x^2/(b-x)

>> [n,d]=numden(f)  % 提取分子和分母
n=
 a*x^2
```

```
d=
 b-x
>> g= ' 3/2*x^2+2/3*x-3/5 '   % 定义符号表达式
g=
 3/2*x^2+2/3*x-3/5

>> [n,d]=numden(g)
n=
 45*x^2+20*x-18
d=
 30

>> h= ' (x^2+3)/(2*x-1)+3*x/(x-1) '
h=
 (x^2+3)/(2*x-1)+3*x/(x-1)

>> [n,d]=numden(h)  %  提取分子分母
n=
 x^3+5*x^2-3
d=
 (2*x-1)*(x-1)
>> k=sym( ' [3/2,(2*x+1)/3;4/x^2,3*x+4] ' )
k=
 [   3/2,(2*x+1)/3]
 [4/x^2,     3*x+4]

>> [n,d]=numden(k)
n=
 [3, 2*x+1]
 [4, 3*x+4]
d=
 [   2,3]
 [x^2,1]
```

【实例讲解】 各个表达式的提取结果都是正确的。在提取各部分之前，3、4 两个表达式 g 和 h 被有理化，并变换成具有分子和分母的一个简单表达式。最后一个表达式 k 是符号数组，numden 返回两个新数组 n 和 d，其中 n 是分子数组，d 是分母数组。如果采

用 s=numden（f）形式，numden 仅把分子返回到变量 s 中。

11.2.2　symadd 函数——符号表达式求和

【语法说明】 symadd(a,b)：返回符号表达式 a 和 b 的和，也可直接用 a+b。

【功能介绍】 符号表达式求和。

【实例 11.2】 求下面两个表达式 $f1=2x^2+3x-5$、$f2=x^2-x+7$ 的和。

```
>> f= ' 2*x^2+3*x-5 '
f=
 2*x^2+3*x-5

>> g= ' x^2-x+7 '
g=
 x^2-x+7

>> symadd(f,g)
ans=
 3*x^2+2*x+2
```

【实例讲解】 使用这个函数，可以进行各种符号表达式的求和运算。

11.2.3　symsub 函数——符号表达式求差

【语法说明】 symsub(a,b)：返回符号表达式 a 和 b 的差，也可直接用 a-b。

【功能介绍】 求符号表达式的差。

【实例 11.3】 求下面两个表达式 $f1=2x^2+3x-5$、$f2=x^2-x+7$ 的差。

```
>> f1= ' 2*x^2+3*x-5 '
F1=
 2*x^2+3*x-5

>> f2= ' x^2-x+7 '
F2=
 x^2-x+7

>> symsub(f1,f2)
```

```
ans=
 x^2+4*x-12
```

【实例讲解】 很容易看出，上面的运算结果是正确的。

11.2.4 symlnul 函数——符号表达式求积

【语法说明】 symmul(a,b)：返回符号表达式 a 和 b 的积，也可直接用 a*b 。

【功能介绍】 求两个符号表达式的积。

【实例 11.4】 求下面两个表达式 $f1=2x^2+3x-5$、$f2=x^2-x+7$ 的积。

```
>> f1= ' 2*x^2+3*x-5 '
f1=
 2*x^2+3*x-5

>> f2= ' x^2-x+7 '
f2=
 x^2-x+7
>> symmul(f1,f2)
ans=
 (2*x^2+3*x-5)*(x^2-x+7)
```

【实例讲解】 符号表达式与数值运算的差别是，不会给出具体的运算结果，而是直观显示的都是符号形式的运算。

11.2.5 symdiv 函数——符号表达式求商

【语法说明】 symdiv(a,b)：返回符号表达式 a 和 b 的商，也可直接用 a/b。

【功能介绍】 求两个符号表达式的商。

【实例 11.5】 求下面两个表达式 $f1=2x^2+3x-5$、$f2=x^2-x+7$ 的商。

```
>>f1= ' 2*x^2+3*x-5 '
f1=
 2*x^2+3*x-5

>> f2= ' x^2-x+7 '
f2=
 x^2-x+7
>> symdiv(f1,f2)
```

```
ans=
 (2*x^2+3*x-5)/(x^2-x+7)
```

【实例讲解】 symdiv 函数可以使符号表达式的运算更加方便。

11.2.6 sympow 函数——符号表达式求幂次

【语法说明】 sympow(a,b)：返回符号表达式 a 的 b 次幂，也可直接用 a^b。

【功能介绍】 求符号表达式的幂次。

【实例 11.6】 求表达式 $f1=2x^2+3x-5$ 的 3 次幂。

```
>>f1= ' 2*x^2+3*x-5 '
f1=
 2*x^2+3*x-5

>> sympow(f1, ' 3*x ' )  % find an expression for f^3*
ans=
 (2*x^2+3*x-5)^3**
```

【实例讲解】 注意函数中幂次参数 b 的写法。

11.2.7 compose 函数——符号的复合函数运算

【语法说明】

- compose(f,g) ：返回复合函数 f(g(y))。
- compose(f,g,z) ：返回自变量为 z 的复合函数 f(g(z))。
- compose(f,g,x,z) ：返回复合函数 f(g(z))，并使 x 成为 f 函数的独立变量。即，如果 f=cos(x/t)，则 compose(f,g,x,z)：返回复合函数 cos(g(z)/t)，而 compose(f,g,t,z)返回 cos(x/g(z))。
- compose(f,g,x,y,z) ：返回复合函数 f(g(z))，并且使 x 与 y 分别成为 f 与 g 函数的独立变量。即如果 f=cos(x/t)，g=sin(y/u)，compose(f,g,x,y,z) 返回 cos(sin(z/u)/t)，而 compose(f,g,x,u,z) 返回 cos(sin(y/z)/t)。

【功能介绍】 求复合函数的运算。

【实例 11.7】 把 $f(x)$和 $g(x)$复合成 $f(g(x))$及 $g(f(x))$，$f(x)$和 $g(x)$分别如下所示：

$$f=\frac{1}{1+x^2} \qquad g=\sin(x)\text{。}$$

```
>> f= ' 1/(1+x^2) '
f =

 1/(1+x^2)

>> g= ' sin(x) '

g =

 sin(x)
>> compose(f,g)
ans=
 1/(1+sin(x)^2)
>> compose(g,f)
ans=
 sin(1/(1+x^2))
```

【实例讲解】 在使用 compose 函数的时候，用户需要注意如何组合。

11.2.8 fincerse 函数——求函数的逆函数

【语法说明】

- finverse(f) ：返回符号函数 f 的反函数。
- finverse(f,v) ：返回自变量为 v 的符号函数 f 的反函数。

【功能介绍】 求符号函数的逆函数。

【实例 11.8】 求下面几个符号表达式 $1/x$、x^2、$ax+b$、$ab+cd-az$ 的逆函数。

```
>>finverse('1/x)%the inverse of 1/x is 1/x since'1/
(1/x)=x'
ans=
 1/x

>> finverse('x^2') % g(x^2)=x has more than one solution
Warning:  finverse(x^2) is not unique
ans=
```

```
    x^(1/2)

    >>finverse('a*x+b') % find the solution to'g(f(x))=x'
    ans=
    -(b-x)/a

    >>finverse('a*b+c*d-a*z'),'a')  %  find  the  solution
to'g(f(a))=a '
    ans=
     -(c*d-a)/(b-z)
```

【实例讲解】 符号表达式的逆函数相当于数值中的逆运算。

11.2.9　symsun 函数——求表达式的符号和

【语法说明】

- symsum(s)：返回 $\sum_{0}^{x-1} s(x)$。
- symsum(s,v)：返回 $\sum_{0}^{x-1} s(v)$。
- symsum(s,a,b)：返回 $\sum_{a}^{b} s(x)$。
- symsum(s,v,a,b)：返回 $\sum_{a}^{b} s(v)$。

【功能介绍】 求表达式符号的和。

【实例 11.9】 求 $\sum_{0}^{x-1} x^2$、$\sum_{1}^{n} (2n-1)^2$、$\sum_{1}^{\infty} \frac{1}{(2n-1)^2}$。

```
>> symsum('x^2')
ans=
    1/3*x^3-1/2* x^2+1/6*x

>> sym('(2*n-1)^2',1,'n')
ans=
    11/3*n+8/3-4*(n+1)^2+4/3*(n+1)^3

>> symsum( ' 1/(2*n-1)^2 ' ,1,inf)
ans=
    1/8*pi^2
```

【实例讲解】 符号运算的结果与数值运算中相同。

11.2.10 sym 函数——数字参量转换为符号表达式

【语法说明】 sym(x)：将数字参数转化为符号表达式。

【功能介绍】 数字变量转化为符号表达式。

【实例 11.10】 把数字 456 转换为符号。

```
>> sym(456)

ans =

456
```

【实例讲解】 这个函数在符号运算的章节中已经详细介绍，这里就不重复了。

11.2.11 numneric 函数——符号表达式转换为数字参量

【语法说明】 numeric(p)：将符号表达式 p 转化为数值表达式。

【功能介绍】 把一个符号常数（无变量符号表达式）变换为一个数值。

【实例 11.11】 把符号表达式(1+sqrt(5))/2 转化为数值表达式。

```
>> rr=' (1+sqrt(5))/2 '
rr=
     (1+sqrt(5))/2
>> numeric(rr)
ans=
     1.6180
>> eval(rr)
ans=
     1.6180
```

【实例讲解】 eval 函数也有同样的效果。

11.2.12 sym2poly 函数——将符号多项式变换成它的 MATLAB 等价系数向量

【语法说明】 sym2poly(p)：将符号多项式 p 转换成它的

MATLAB 等价系数向量。

【功能介绍】 将符号多项式变换成它的MATLAB等价系数向量。

【实例 11.12】 sym2poly 用来得到多项式等价系数向量，通过 poly2sym 命令可以还原，如果在 poly2sym(n，' s ')中指定了自变量，那么得到的多项式就是关于这个自变量的，如“s”。

```
>> f=' 2*x^2+x^3-3*x+5 '
f=
     2*x^2+x^3-3*x+5

>> n=sym2poly(f)
n=
     1   2   -3   5
>> poly2sym(n)
ans=
    2*x^2+x^3-3*x+5

>> poly2sym(n,' s ')
ans=
    s^3+2*s^2-3*s+5
```

【实例讲解】 使用上面的函数，可以很方便地获取系统的系数。

11.2.13　subs 函数——变量替换

【语法说明】 subs(f,new,old)：f 为符号表达式，new 与 old 是字符、字符串或其他的符号表达式，new 字符串将替换符号表达式 f 中的 old 字符串。

【功能介绍】 将符号表达式中的变量进行替换。

【实例 11.13】 对符号表达式 ax^2+bx+c 中的变量和系数进行替换。

```
>> f= ' a*x^2+b*x+c '    %  建立函数
f=
     a*x^2+b*x+c

>> subs(f,' s ',' x ')  %  用“s”代替 ' x '
ans=
     a*s^2+b*s+c
```

```
>> subs(f,' bate ',' a ')  % 用 ' bate '  代替  ' a '
ans=
     bate*x^2+b*x+c

>> g=' 3*x^2+5*x-4 '
g=
     3*x^2+5*x-4

>> h=subs(g,' 2 ',' x ')  %  代入常数
h=
     18
```

【实例讲解】 用 s 代替 x 后，得到的函数式就变成以 s 为自变量的表达。其余类似。

11.2.14　digit 函数——可变精度算数运算

【语法说明】

- digit：查看目前系统中的算术运算精度。
- digit(n)：将系统的运算精度调整为小数点后 n 位。

【功能介绍】 调整系统的运算精度。

【实例 11.14】 通过指定系统的精度位数，来观察一些数据的精度。

```
>> pi

ans =

    3.1416
>> format  long  %  指定精度

>> pi  %  how about π to numeric accuracy
ans=
    3.14159265358979

>> digits          %  显示系统的精度
Digits=16

>> vpa(' pi ')    %  按照系统精度的pi值
```

```
ans=
    3.141592653589793

>> digits(18)        % 改变精度为 18

>> vpa(' pi ')
ans=
    3.14159265358979324

>> vpa( ' pi ' ,20)
ans=
    3.1415926535897932385

>> vpa( ' pi ' ,50)
ans=
   3.1415926535897932384626433832795028841971693993751

>> vpa( ' 2^(1/3) ' ,200)
ans=
1.25992104989487316476721060727822835057025146470150
798008197511215529967651
1395948372939656243625509415431025603561566525939900
240406137372284591103042
693552469606426166250009774745265654803068671854055
```

【实例讲解】 以上实例演示了在规定精度之后的数值表示。

11.3 符号方程求解

用 MATLAB 本身具有的符号函数可以求解符号方程。求解的方法和实例将在下面这节进行详细的介绍。

11.3.1 solve 函数——求解线性符号方程组

【语法说明】

- solve(f)：求解线性符号方程 f。
- solve(f,g)：求解线性符号方程组 f、g。

【功能介绍】 求解线性符号方程组。

【实例 11.15】 求解下面几个表达式和方程。

```
>> solve( ' a*x^2+b*x+c ' )  %求解这个线性方程组
ans=
     [1/2/a*(-b+(b^2-4*a*c)^1/2)]
     [1/2/a*(-b-(b^2-4*a*c)^1/2)]

>> solve( ' a*x^2+b*x+c ' , ' b ' )  %求出 b 的值
ans=
     -(a*x^2+c)/x
>> f=solve( ' cos(x)=sin(x) ' )     % 解方程
f=
     1/4*pi

>> t=solve( ' tan(2*x)=sin(x) ' )
t=
     [                  0]
     [acos(1/2+1/2*3^(1/2))]
     [acos(1/2=1/2*3^(1/2))]
```

求得数值解：

```
>> numeric(f)
ans=
     0.7854

>> numeric(t)
ans=
     0
     0 + 0.8314i
    1.9455
```

【实例讲解】

- 对于第一个小例子，表达式不是一个方程式（不含等号），则在求解之前函数 solve 将表达式置成 0。
- 对于第二个例子，结果是符号向量，其元素是方程的 2 个解。如果想对非缺省 x 变量求解，solve 必须指定变量。
- 在求解周期函数方程时，有无穷多的解。在这种情况下，solve 对解的搜索范围限制在接近于零的有限范围，并返回非唯一的解的子集。

11.3.2 代数方程组求解

【语法说明】

- solve(s1，s2，...，sn)：对缺省变量求解具有 n 个方程的方程组。
- solve(s1，s2，...，sn，'v1，v2，...，vn')：对 n 个 v1，v2，…，vn 的未知数求解具有 n 个方程的方程组。

【功能介绍】 求解代数方程组。

【实例 11.16】 求解下面这个代数方程组。

$$x+\frac{y+z}{2}=w \quad w=y+x+w-10 \quad w+x=z+\frac{y}{4} \quad w+z=y+8x-1$$

```
>> Y1= ' x+(y+z)/2=w ' ;

>> Y2= ' z=y+x+w-10 ' ;

>> Y3= ' w+x=z+y/4 ' ;

>> Y4= ' w+z=y+8*x-1 ' ;

>>[X1,X2,X3,X4]=solve(Y1,Y2,Y3,Y4,' z,y,x,w ' )
X1=
    16
X2=
    8
X3=
    3
X4=
    15
```

【实例讲解】 求得的 4 个解是 4 个方程中 4 个未知数的值。

11.3.3 dsolve 函数——符号微分方程求解

【语法说明】 dsolve('eqn1', 'eqn2',…)：求解符号微分方程，参数 eqn1,eqn2…代表微分方程与初始条件。

【功能介绍】 求解符号微分方程。

【实例 11.17】 求解 $dy/dx=1+y^2$ 并且当 y(0)=1 时的解。

```
>> dsolve(' Dy=1+y^2 ',' y(0)=1 ')
y=
    tan(x+1/4*pi)
```

【实例讲解】 最后给出的结果是具有周期性的。

【实例 11.18】 求解二阶微分方程 $\frac{d^2y}{dx^2}=\cos(2x)-y$，初始条件为 $\frac{dy}{dx}(0)=0$ $y(0)=1$。

```
>> y=dsolve(' D2y=cos(2*x)-y ',' Dy(0)=0 ',' y(0)=1 ')
y=
    -2/3*cos(x)^2+1/3+4/3*cos(x)

>> y=simple(y)  %  y looks like it can be simplified
y=
    -1/3*cos(2*x)+4/3*cos(x)
```

【实例讲解】 用户可以使用微分方程的知识检测上面的结果。

【实例 11.19】 求解两个微分方程式，下面有两个线性一阶方程，如下：

$\frac{dy}{dx}=3f+4g$ $\frac{dg}{dx}=-4f+3g$ 其初始条件为：$f(0)=0$ 和 $g(0)=1$。

通解求解的结果为：

```
>> [f,g]=dsolve( ' Df=3*f+4*g ' , ' Dg=-4*f+3*g ' )
f=
    C1*exp(3*x)*sin(4*x)+C2*exp(3*x)*cos(4*x)
g=
    -C2*exp(3*x)*sin(4*x)+C1*exp(3*x)*cos(4*x)
```

加上初始条件：$f(0)=0$ 和 $g(0)=1$，可以得到：

```
>>[f,g]=dsolve('Df=3*f+4*g','Dg=-4*f+3*g','f(0)=0,g(
0)=1')
f=
    exp(3*x)*sin(4*x)
g=
    exp(3*x)*cos(4*x)
```

【实例讲解】 加上初始条件后就得到了微分方程的特解。

11.3.4 diff 函数——符号函数微分

【语法说明】

- diff(f) ：返回 f 的微分。
- diff(f, 'a') ：对 a 变量求微分。
- diff(f,n) ：对 f 求 n 次微分。
- diff(f, 'a',n)： 对变量 a 求 n 次微分。

【功能介绍】 求符号函数的微分。

【实例 11.20】 对函数式 ax^3+bx^2+cx+d 求微分。

```
>>f='a*x^3+b*x^2-c*x-d' % define a symbolic expression
f=
     a*x^3+b*x^2-c*x-d

>>diff(f) % differentiate with respect to the default
variable x
ans=
     3*a*x^2+2*b*x-c

>> diff(f,'a ')  %  differentiate with respect to a
ans=
     x^3

>>diff(f,2) % differentiate twice with respect to x
ans=
     6*a*x+2*b

>>diff(f,'a',2) % differentiate twice with respect to a
ans=
     0
```

【实例讲解】 当对其中一个参数求微分时，其他的参数都作为常数处理。

【实例 11.21】 对数组 $A=\begin{bmatrix} ax & bx^2 \\ cx^3 & dy \end{bmatrix}$ 进行微分。

```
>> F=sym(' [a*x,  b*x^2;  c*x^3,  d*y] ')
F=
    [   a*x,  b*x^2]
```

```
    [c*x^3,     d*y]

>>diff(F) % differentiate the element with respect to x
ans=
    [          a,2*b*x]
    [3*c*x^2,       0]
```

【实例讲解】 对数组微分相当于对数组中的每一个元素都微分。

11.3.5 int 函数——符号函数积分

【语法说明】

- int(f) ：对 f 求不定积分。
- int(f,v) ：对 v 变量求不定积分。
- int(f,a,b)： 对 f 求[a,b]上的定积分。
- int(f,v,a,b) ：对变量 v 求[a,b]上的定积分。

【功能介绍】 对符号函数进行积分。

【实例 11.22】 对如下的几个简单函数微分。

```
>> f=' sin(s+2*x) ')
f=
     sin(s+2*x)

>> int(f)
ans=
     -1/2*cos(s+2*x)

>> int(f,' s ')
ans=
     -cos(s+2*x)

>> int(f,pi/2,pi)
ans=
     -cos(x)

>> int(f,' s ',pi/2,pi)
ans=
     cos(2*x)-sin(2*x)
```

```
>> int(f,' m ',' n ')
ans=
    -1/2*cos(s+2*n)+1/2*cos(s+2*m)
```

【实例讲解】 用户可以使用微积分的相关知识检测结果的正确性。

【实例 11.23】 对符号数组函数式积分。

```
>>F=sym('[a*x,b*x^2;c*x^3,d*s]') %create a symbolic array
F=
      [ a*x,b*x^2]
      [c*x^3,d*s]

>>diff(F)%ubtegrate the array elements with respect to x
ans=
      [1/2*a*x^2,1/3*b*x^3]
      [1/4*c*x^4,    d*s*x]
```

【实例讲解】 对于符号数组进行积分，相当于对数组中的每一个元素进行积分。

11.3.6　ezplot 函数——符号表达式画图

【语法说明】

- ezplot(f)：在默认区间-2*pi<x<2*pi 绘制 $f=f(x)$的函数图。
- ezplot(f,[a,b])：　在区间 a<x<b 上绘制 $f=f(x)$的函数图。

【功能介绍】 对符号函数进行画图。

【实例 11.24】 绘制函数 $y=16x^2+64x+96$。

```
>>y='16*x^2+64*x+96 ' % expression to plot
y=
     16*x^2+64*x+96
>> ezplot(y)
>> ezplot(y,[0 6])
```

得到的结果如图 11.1 和图 11.2 所示。

【实例讲解】 ezplot 绘制了定义域为$-2\pi \leqslant x \leqslant 2\pi$的给定符号函数，并相应地调整了 y 轴比例，还加了网格栅和标志。而如果函数限定了区间为[0，6]。

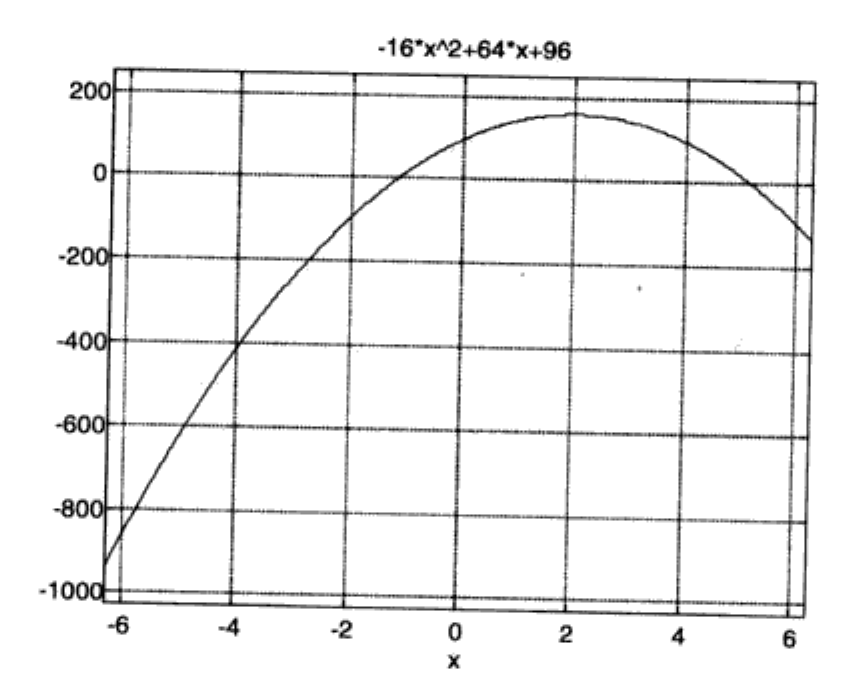


图 11.1 符号函数 16*x^2+64*x+96 -6≤x≤6

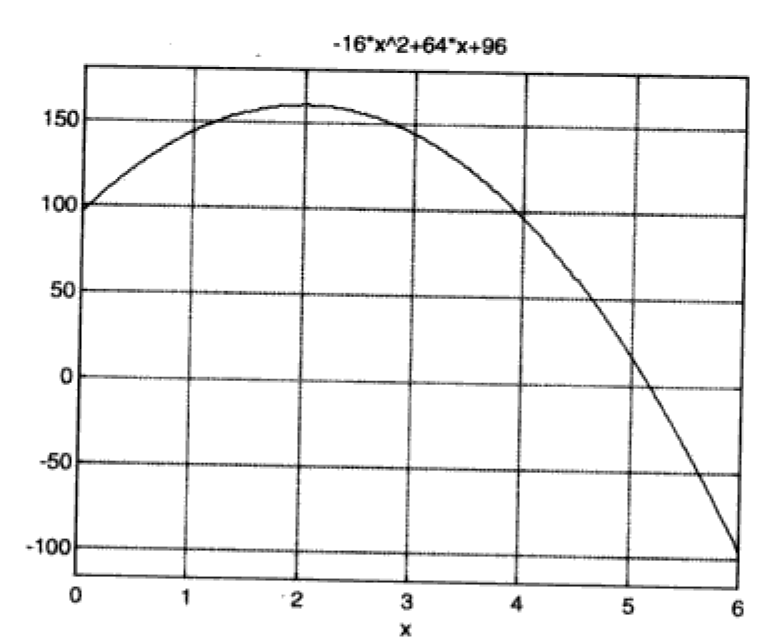


图 11.2 符号函数 16*x^2+64*x+96 0≤x≤6

11.3.7 pretty 函数——符号函数化简

【语法说明】

- pretty(f) ：将符号表达式化简成与高等数学课本上显示符号表达式形式类似。
- collect(f) ：合并符号表达式的同类项。
- horner(f) ：将一般的符号表达式转换成嵌套形式的符号表达式。

【功能介绍】 将符号函数简化或格式化。

【实例 11.25】 将泰勒级数进行简化。

```
>> f=taylor( ' log(x+1)/(x-5) ' )  %
f =
    -1/5*x+3/50*x^2-41/750*x^3+293/7500*x^4-1207/37500*x^5+O(x^6)

>> pretty(f)
                        2    41   3   293    4   1207    5    6
           -1/5x+3/50x - -----x + ------x - -------x +O(x)
                              750       7500       37500
```

【实例讲解】 简化的实质是利用了截断误差，即保留部分项。

【实例 11.26】 用不同的简化形式来简化函数式。

```
>> f=sym('(x^2-1)*(x-2)*(x-3)')  % create a function
f=
    (x^2-1)*(x-2)*(x-3)

>> collect(f)
ans=
    x^4-5*x^3+5*x^2+5*x-6

>> hornor(ans)
ans=
    -6+(5+(5+(-5+x)*x)*x)*x

>> factor(ans)
ans=
    (x-1)*(x-2)*(x-3)*(x+1)

>> expand(f)
ans=
    x^4-5*x^3+5*x^2+5*x-6
```

【实例讲解】 collect 和 expand 函数将函数式展开，hornor 函数按另一种方式进行整理各项。

11.3.8 simplify 函数——利用恒等式化简

【语法说明】 simplify(f)：对符号表达式进行化简，它利用各种类型的代数恒等式，包括求和、积分、三角函数、指数函数以及 Bessel 函数等来化简符号表达式。

【功能介绍】 利用数学中的恒等式化简。

【实例 11.27】 化简下面几个简单函数表达式。

```
>> simplify('log(2*x/y)')
ans=
    log(2)+log(x)-log(y)

>> simplify('sin(x)^2+3*x+cos(x)^2-5')
ans=
    3*x-4
```

```
>> simplify( ' (-a^2+1)/(1-a) ' )
ans=
    a+1
```

【实例讲解】 利用数学恒等式来进行简化，大大减少了工作量。

11.3.9 simple 函数——最少字符简化

【语法说明】

- simple(f)：对符号表达式尝试多种不同的算法进行化简，以显示长度最短的符号表达式简化形式。
- [r,how]=simple(f) ：返回的 r 为符号表达式进行化简后的形式，how 为所采用的简化方法。

【功能介绍】 尝试多种简化方法，最后选择最短的符号表达方式。

【实例 11.28】 对符号表达式 $f=\sqrt[3]{\frac{1}{x^3}+\frac{6}{x^2}+\frac{12}{x}+8}$ 进行简化。

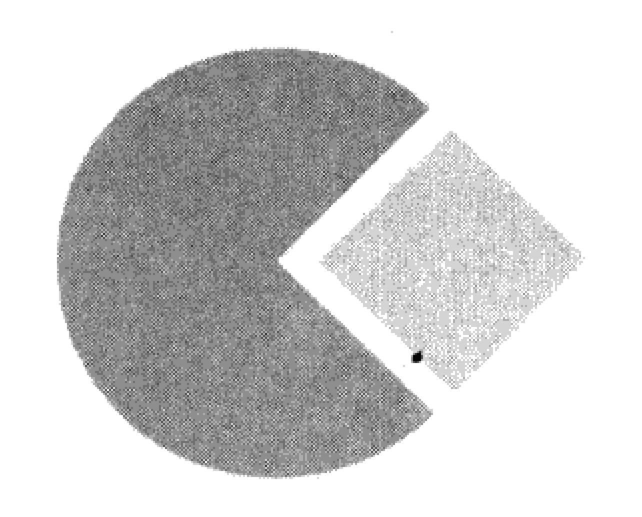
```
>> f=' (1/x^3+6/x^2+12/x+8)^(1/3) '  % 建立一个表达式
f=
    (1/x^3+6/x^2+12/x+8)^(1/3)

>> simple(f)  % simplify it
simplify :
    (2*x+1)/x
ans=

>> simple(ans)  % 对(2*x+1)/x 进行化简

combine(trig)
    2+1/x
ans=
    2+1/x
```

【实例讲解】 从上面的结果中可以看出，符号简化的结果符合数学公式。



附录　MATLAB 常用函数检索表（按首字母排序）

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z